

JC960 U.S. PTO
11/28/00

11-30-00

A

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Priority Application Serial No.09/040,813
Priority Filing Date3/18/1998
Inventor.....Misra et al.
Applicant Microsoft Corporation
Priority Group Art Unit.....2786
Priority Examiner Elmore, R
Attorney's Docket No.MS1-197USC1
Title: System and Method for Software Licensing

JC930 U.S. PTO
09/724703
11/28/00

CONTINUATION PATENT APPLICATION TRANSMITTAL

To: Commissioner of Patents and Trademarks,
Washington, D.C. 20231

From: Allan T. Sponseller (Tel. 509-324-9256; Fax 509-323-8979)
Lee & Hayes, PLLC
421 W. Riverside Avenue, Suite 500
Spokane, WA 99201

The following enumerated items accompany this transmittal letter and are being submitted for the matter identified in the above caption.

1. Transmittal Letter with Certificate of Mailing included.
2. Fee Transmittal
3. Check in the Amount of \$2434.00
4. PTO Return Postcard Receipt
5. Request for Continuation Application Under 37 C.F.R. 1.53 (b)
6. True copy of the prior application as filed including the specification (57 pages, claims 1-78 & Abstract)
7. 8 Sheets Formal Drawings (Figs. 1-8)
8. Copy of Executed Declaration
9. Preliminary Amendment
10. Information Disclosure Statement, including form PTO-1449
11. Request to Approve Drawing Changes including photocopy of Fig. 3 with proposed changes in red
12. Copy of Assignment

Large Entity Status ☒ Small Entity Status ☐

The Commissioner is hereby authorized to charge payment of fees or credit overpayments to Deposit Account No. 12-0769 in connection with any patent application processing fees under 37 CFR 1.17; and any additional filing fees under 37 CFR 1.16 for the presentation of extra claims.

EL685270532

PTO/SB/17 (11-00)

Approved for use through 10/31/2002 OMB 0651-0032

U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

**FEE TRANSMITTAL
for FY 2001**

Patent fees are subject to annual revision.

Complete if Known

Application Number

Filing Date

First Named Inventor

MSra

Examiner Name

Group Art Unit

Attorney Docket No.

MSI-191USCI

TOTAL AMOUNT OF PAYMENT

(\$) 2434.00

METHOD OF PAYMENT

- 1.
- ☒
- The Commissioner is hereby authorized to charge indicated fees and credit any overpayments to.

Deposit
Account
Number

12-0769

Deposit
Account
Name

Lee & Hayes, PLLC

- ☒
- Charge Any Additional Fee Required
-
- Under 37 CFR 1.16 and 1.17

- ☐
- Applicant claims small entity status.
-
- See 37 CFR 1.27

- 2.
- ☒
- Payment Enclosed:**

- ☒
- Check
- ☐
- Credit card
- ☐
- Money
-
- Order
- ☐
- Other

FEE CALCULATION**1. BASIC FILING FEE**

Large Entity Small Entity

Fee Fee Fee Fee Fee Description

Code (\$) Code (\$) Code (\$)

101 710 201 355 Utility filing fee

106 320 206 160 Design filing fee

107 490 207 245 Plant filing fee

108 710 208 355 Reissue filing fee

114 150 214 75 Provisional filing fee

Fee Paid

710

SUBTOTAL (1) (\$) 710

2. EXTRA CLAIM FEES

Total Claims 58 -20** = 38 X 18 = 684

Independent Claims 16 - 3** = 13 X 80 = 1040

Multiple Dependent _____ = _____

Large Entity Small Entity

Fee Fee Fee Fee Fee Description

Code (\$) Code (\$) Code (\$)

103 18 203 9 Claims in excess of 20

102 80 202 40 Independent claims in excess of 3

104 270 204 135 Multiple dependent claim, if not paid

109 80 209 40 ** Reissue independent claims

over original patent

110 18 210 9 ** Reissue claims in excess of 20

and over original patent

SUBTOTAL (2) (\$) 1724

**or number previously paid, if greater; For Reissues, see above

FEE CALCULATION (continued)**3. ADDITIONAL FEES**

Large Entity Small Entity

Fee Fee Fee Fee Fee Description

Code (\$) Code (\$) Code (\$)

105 130 205 65 Surcharge - late filing fee or oath

127 50 227 25 Surcharge - late provisional filing fee or

cover sheet

139 130 139 130 Non-English specification

147 2,520 147 2,520 For filing a request for ex parte reexamination

112 920* 112 920* Requesting publication of SIR prior to

Examiner action

113 1,840* 113 1,840* Requesting publication of SIR after

Examiner action

115 110 215 55 Extension for reply within first month

116 390 216 195 Extension for reply within second month

117 890 217 445 Extension for reply within third month

118 1,390 218 695 Extension for reply within fourth month

128 1,890 228 945 Extension for reply within fifth month

119 310 219 155 Notice of Appeal

120 310 220 155 Filing a brief in support of an appeal

121 270 221 135 Request for oral hearing

138 1,510 138 1,510 Petition to institute a public use proceeding

140 110 240 55 Petition to revive - unavoidable

141 1,240 241 620 Petition to revive - unintentional

142 1,240 242 620 Utility issue fee (or reissue)

143 440 243 220 Design issue fee

144 600 244 300 Plant issue fee

122 130 122 130 Petitions to the Commissioner

123 50 123 50 Processing fee under 37 CFR 1.17(q)

126 180 126 180 Submission of Information Disclosure Stmt

581 40 581 40 Recording each patent assignment per

property (times number of properties)

146 710 246 355 Filing a submission after final rejection

(37 CFR § 1.129(a))

149 710 249 355 For each additional invention to be

examined (37 CFR § 1.129(b))

179 710 279 355 Request for Continued Examination (RCE)

169 900 169 900 Request for expedited examination

of a design application

Other fee (specify) _____

*Reduced by Basic Filing Fee Paid

SUBTOTAL (3) (\$)

SUBMITTED BY

Name (Print/Type)

Allan J. Sponseller

Registration No.
(Attorney/Agent)

38 318

Complete (if applicable)

Telephone

(509) 324-9250

Signature

[Signature]

Date

11/28/00

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application Serial No.Not Assigned Yet
 Filing DateNovember 28, 2000
 Inventor..... Pradyunma K. Misra et al.
 Group Art UnitNot Assigned Yet
 ExaminerNot Assigned Yet
 Attorney's Docket No.MS1-197USC1
 Title: System and Method for Software Licensing

PRELIMINARY AMENDMENT

To: Commissioner of Patents and Trademarks,
 Washington, D.C. 20231

From: Allan T. Sponseller (Tel. 509-324-9256; Fax 509-323-8979)
 Lee & Hayes, PLLC
 421 W. Riverside Ave., Suite 500
 Spokane, WA 99201

In the Specification:

Please amend the specification as follows:

On page 1, line 6, please delete the first occurrence of "been".

On page 18, line 24, please change "be" to --been--.

On page 31, line 25, please delete "a".

On page 35, line 8, please change "ganeration" to --generation--.

Please insert the following section at the beginning of the specification
 before the heading "Technical Field:"

--RELATED APPLICATIONS

This is a continuation of U.S. Patent Application Serial No. 09/040,813,
 filed March 18, 1998.--

1
2 **In the Claims:**

3 Please cancel claims 7, 8, 9, 11, 12, 22, 28, 37, 38, 39, 40, 41, 42, 43, 44,
4 58, 62, 66, 67, and 70 without prejudice.

5 Claims 1-6, 10, 13-21, 23-27, and 29-36 are amended.

6 Claims 1-6, 10, 13-21, 23-27, 29-36, 45-57, 59-61, 63-65, 68, 69, and 71-78
7 are pending.

8
9 Please amend claims 1-6, 10, 13-21, 23-27, and 29-36 as follows:
10

11
12 1. (Amended) A computer-implemented method comprising [the
13 following steps]:

14 creating a license pack at a license generator, the license pack containing a
15 set of one or more individual software licenses;

16 signing the license pack with a digital signature of the license generator;

17 issuing the license pack to a license server;

18 verifying, at the license server, the license generator's digital signature on
19 the license pack; and

20 distributing the software licenses contained in the license pack from the
21 license server to corresponding clients.
22

23 2. (Amended) A computer-implemented method as recited in claim 1,
24 further comprising [the step of] creating a license pack containing a predefined
25 number of software licenses.

1
2 3. (Amended) A computer-implemented method as recited in claim 1,
3 further comprising [the following steps]:

4 creating a license pack ID at the license generator; and
5 evaluating the license pack ID at the license server.
6

7 4. (Amended) A computer-implemented method as recited in claim 1,
8 further comprising [the following steps]:

9 encrypting the license pack at the license generator; and
10 decrypting the license pack at the license server.
11

12 5. (Amended) A computer-implemented method as recited in claim 1,
13 further comprising [the step of] creating a license pack that is tailored to a
14 particular operating platform of the clients.
15

16 6. (Amended) A computer-implemented method as recited in claim 1,
17 further comprising [the step of] determining an authenticity of an individual client
18 prior to distributing the software license to that individual client.
19

20 10. (Amended) A computer-implemented method as recited in claim 1,
21 wherein the license pack has a license pack ID, and further comprising [the step
22 of] granting additional licenses for the license pack having the same license pack
23 ID.
24
25

1 13. (Amended) A computer-implemented method for distributing
2 software licenses to clients so that the clients may legally execute underlying
3 software to which the software licenses pertain, the computer-implemented
4 method comprising [the step of] electronically issuing the software licenses as
5 digital certificates that can be distributed in one-to-one correlation with individual
6 clients and traced to an issuing authority.

7
8 14. (Amended) A computer-readable medium having computer readable
9 instructions for performing the [step] method as recited in claim 13.

10
11 15. (Amended) A computer-implemented method comprising [the
12 following steps]:

13 receiving a request for a software license from a particular license server;
14 creating a license pack containing a set of one or more individual software
15 licenses;
16 assigning a license pack ID to the license pack;
17 associating the license pack ID with the particular license server;
18 digitally signing the license pack; and
19 issuing the signed license pack to the particular license server.

20
21 16. (Amended) A computer-implemented method as recited in claim 15,
22 further comprising [the step of] creating a license pack containing a predefined
23 number of software licenses.

1 17. (Amended) A computer-implemented method as recited in claim 15,
2 further comprising [the step of] creating a license pack that includes a platform
3 type indicating a type of operating platform for which the software licenses can be
4 used.

5
6 18. (Amended) A computer-implemented method as recited in claim 15,
7 further comprising [the step of] creating a license pack that includes a predefined
8 number of software licenses, a platform type indicating a type of operating
9 platform for which the software licenses can be used, an expiration date indicating
10 a date on which the software licenses will expire, and a product ID that identifies a
11 product with which the software licenses can be used.

12
13 19. (Amended) A computer-implemented method as recited in claim 15,
14 further comprising [the step of] encrypting the license pack.

15
16 20. (Amended) A computer-readable medium having computer readable
17 instructions for performing the [steps] method as recited in claim 15.

18
19 21. (Amended) A computer-implemented method comprising [the
20 following steps]:

21 receiving a request for a software license from a particular client;

22 determining an authenticity of the particular client;

23 selecting a software license from a pack of software licenses that is
24 appropriate for the particular client, the software license having an associated
25 license ID;

1 associating the license ID with the particular client; and
2 granting the software license to the particular client.
3

4 23. (Amended) A computer-implemented method as recited in claim 21,
5 further comprising [the step of] granting the software license as containing the
6 license ID, a platform type indicating a type the platform, an issue date indicates a
7 date on which the license is issued to the client, an expiration date that indicates a
8 date on which the software license will expire, a product ID that identifies a
9 product with which the software licenses can be used, a client ID that identifies the
10 particular client, and a version of the software license.
11

12 24. (Amended) A computer-implemented method as recited in claim 21,
13 wherein [the step of] determining the authenticity comprises [the following steps]:
14 receiving a client software ID from the particular client; and
15 evaluating the client software ID to determine whether the client is
16 authentic.
17

18 25. (Amended) A computer-implemented method as recited in claim 21,
19 wherein [the step of] determining the authenticity comprises [the following steps]:
20 maintaining a set of client images;
21 receiving a client software ID from the particular client; and
22 comparing the client software ID to the client images to evaluate whether
23 the client is authentic.
24
25

1 26. (Amended) A computer-implemented method as recited in claim 21,
2 further comprising [the following steps]:
3 determining a platform of the particular client; and
4 selecting the software license as is appropriate for the platform of the
5 particular client.

6
7 27. (Amended) A computer-implemented method as recited in claim 21,
8 further comprising [the step of] encrypting the software license using a public key
9 of the particular client.

10
11 29. (Amended) A computer-readable medium having computer readable
12 instructions for performing the [steps] method as recited in claim 21.

13
14 30. (Amended) A computer-implemented method comprising [the
15 following steps]:

16 computing, at a computer, a value as a one-way function of a client
17 executable image that uniquely identifies a client; and

18 digitally signing the value, at the computer, using a private signing key of a
19 server that serves the client to create a client image digital signature that is unique
20 to the client.

21
22 31. (Amended) A computer-implemented method as recited in claim 30,
23 wherein [the] computing the value [step] comprises [the step of] hashing the
24 executable image to produce a hash value.

1 32. (Amended) A computer-implemented method as recited in claim 30,
2 further comprising [the step of] storing the client image digital signature at the
3 client.

4
5 33. (Amended) A computer-implemented method as recited in claim 30,
6 further comprising [the following steps]:

7 storing the client executable image at the server;

8 storing the client image digital signature at the client;

9 submitting the client image digital signature from the client to the server
10 when requesting a software license; and

11 evaluating an authenticity of the client based on the client image digital
12 signature prior to granting a software license to the client
13

14 34. (Amended) A computer-implemented method as recited in claim 33,
15 wherein [the] evaluating the authenticity [step] comprises [the following steps]:

16 unsigning the client image digital signature using a public key of the server
17 to recover the client executable image; and

18 comparing the recovered client executable image to the client executable
19 image stored at the server.
20

21 35. (Amended) A computer-implemented method as recited in claim 34,
22 further comprising [the step of] rejecting the request for a software license in an
23 event that the recovered client executable image does not match the client
24 executable image stored at the server.
25

1 36. (Amended) A computer-readable medium having computer readable
2 instructions for performing the [steps] method as recited in claim 30.

3
4
5
6 REMARKS

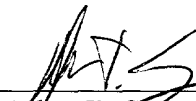
7 Applicant respectfully requests entry of this amendment before examination
8 of the subject application.

9 In an Office Action of the priority application (U.S. Patent Application
10 Serial No. 09/040,813), several claims were rejected under 35 U.S.C. §103(a) by a
11 combination of references including U.S. Patent No. 5,790,677 to Fox et al. Given
12 that the filing date of the subject application is after November 29, 1999,
13 Applicant respectfully submits that the Fox et al. patent, which is assigned to the
14 same Assignee as the subject application, is not a useable prior art reference under
15 35 U.S.C. §103(a) for the subject application.

16 Applicant respectfully requests prompt issuance of the subject application.
17

18 Respectfully Submitted,

19 Date: 11/28/00

20 By: 
21 Allan T. Sponseller
22 Reg. No. 38,318
23 (509) 324-9256
24
25

EM044522632

EL685270532

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

System and Method for Software Licensing

Inventor(s):

Pradyunma K. Misra

Bradley J. Graziadio

Terence Spies

ATTORNEY'S DOCKET NO. MS1-197US

1 **TECHNICAL FIELD**

2 This invention relates to systems and methods for licensing software. This
3 invention further relates to systems and methods for enforcing software licenses.
4

5 **BACKGROUND**

6 Software licensing has been historically based on a "trust" model in
7 which the user (i.e., licensee) is presumed to be honest and trustworthy and to
8 abide by the legal requirements of the license. Under the trust model, a software
9 license typically accompanies a software product to explain the terms of use. For
10 instance, the software license might dictate that the program code is to be installed
11 on only one computer, and may be used to make one backup copy.

12 Common types of licenses include "shrink wrap" licenses, "online"
13 licenses, and "site" licenses. A "shrink wrap" license is a license that accompanies
14 each software product that is sold individually in a shrink-wrapped package
15 through retail stores. The user is typically assumed to accept the terms of the
16 shrink wrap license upon breaking the seal of the package, or the container that
17 holds the disk itself.

18 An "online" license is one that accompanies software products that are
19 downloaded online, such as from the Internet. The license is typically presented to
20 the user prior to downloading the code. The user is presented with a choice to
21 accept or reject the license. If the user accepts the license (e.g., by clicking an
22 "Accept" button on the screen), the user is presumed to have accepted the terms of
23 the license and the code is downloaded to the user's computer.

24 A "site" license is a single license that allows installation of multiple copies
25 of software on many different computers at a particular site or many sites. It is

commonly used to sell software to corporations, firms, or other entities having many computers. The purchaser pays for a certain number of copies (e.g., hundreds or thousands), and the site license enables the purchaser to install that many copies on its computers. The site license is beneficial because the software vendor need not supply a large number of program disks, but merely supplies one or a few copies of the software and lets the purchaser install the copies without violating the agreement.

Each of the above license arrangements assumes that the purchaser is honest. The software purchaser must abide by the license terms in order to legally use the software. If the purchaser fails to abide by the provisions, the purchaser can be charged with civil and criminal violations.

However, enforcement of such licenses is impractical, if not impossible. Unscrupulous users might make multiple copies of the software code and install it on more computers than the license allows. Yet, software vendors cannot begin to monitor these abuses because they occur in the privacy of the home or company. Thus, it is believed that the software industry loses a large percentage of revenues each year simply due to illegitimate use of software by the licensees. This loss does not even account for the problems of overseas pirating.

Another problem with conventional software licensing practices concerns internal monitoring and bookkeeping on the part of large-site licensees. In most cases, the licensees want to comply with the terms of the software licenses, but are unable to adequately track the software as it is used throughout the site. For example, a large corporation might purchase several thousand copies of the software and begin installing the copies. However, computers and personnel change over time and it is difficult to centrally monitor how many copies have

1 been installed, whether the copies have expired, whether they need upgrading, and
2 so forth.

3 Accordingly, there is a need to develop a new approach to licensing
4 software in a manner that assures that the terms are being met and assists the
5 licensee with monitoring whether it is in compliance with the software license.

6 7 SUMMARY

8 This invention concerns a system and method for licensing software. The
9 system and method provides confidence to the vendor that the software license is
10 being complied with, while also assisting the purchaser in monitoring its own
11 compliance with the license.

12 According to one aspect of this invention, computer software licenses are
13 electronically issued as digital certificates that can be distributed in one-to-one
14 correlation with individual client computers and traced to an issuing authority.

15 According to another aspect, the system includes a license generator located
16 at a licensing clearinghouse and at least one license server and multiple clients
17 located at or affiliated with a company or other entity. Because the clients might
18 not have network connectivity to the license server, one or more intermediate
19 servers may act as an intermediary for the clients. These intermediate servers are
20 otherwise common servers that provide resources to clients, but with the added
21 ability to facilitate connectivity to the license server for purposes of distributing
22 software licenses to the clients.

23 When a company wants a software license, it sends a purchase request (and
24 an appropriate fee) to the licensing clearinghouse. The license generator at the
25 licensing clearinghouse creates a license pack containing a set of one or more

00327-204260

1 individual software licenses. To prevent the license pack from being copied and
2 installed on multiple license servers, the license generator assigns a unique license
3 pack ID to the license pack and associates the license pack ID with the license
4 server in a master license database kept at the licensing clearinghouse. The license
5 generator also digitally signs the license pack and encrypts it with the license
6 server's public key. The license generator sends the license pack to the license
7 server using standard communications, such as over a data communication
8 network (e.g., Internet) or via a portable data medium (e.g., floppy diskette, CD-
9 ROM, etc.).

10 The license server verifies the license generator's digital signature on the
11 license pack and if valid, installs the license pack for subsequent distribution of
12 licenses. The license server maintains an inventory of software licenses that have
13 been purchased from the licensing clearinghouse. The license server is responsible
14 for distributing the software licenses contained in the license pack to individual
15 clients. It monitors the software licenses that have been granted to clients and
16 continues to distribute licenses as long as non-assigned licenses remain available.
17 Once the supply of non-assigned licenses is exhausted, however, the license server
18 can no longer grant licenses to the clients and the customer must purchase a new
19 pack from the license clearinghouse.

20 When a client connects to a server, the client presents a valid license (if it
21 has one). If the client does not have an appropriate license, the server assists the
22 client in obtaining a license from the license server. This provides an automated
23 mechanism for clients to obtain and license server to distribute licenses to clients.

24 When a license is requested, the license server initially checks if the
25 requesting client has already been issued a license. When this situation is detected,

1 the license server issues the existing license to the client. This is actually reissuing
2 of the same license that was previously issued. This allows the client to gracefully
3 recover licenses when they are lost.

4 In one implementation, the license server determines an appropriate type of
5 license based in part on the client's operating system platform. The license server
6 derives the platform information by establishing a trust relationship with the client
7 and then querying its platform type. If a software license is available for
8 allocation, the license server grants a software license that is appropriate for the
9 client's platform.

10 To prevent an issued license from being copied from one client machine to
11 another, the software license is assigned to a specific client by including its client
12 ID within the license. The software license also has a corresponding license ID
13 that is associated with the client ID in a database record kept at the license server.

14 The license server digitally signs the software license. The license is passed
15 to the client, where it is stored in a local cache at the client. Once a client has
16 obtained a license, it is responsible for managing the storage of that license.

17 18 **BRIEF DESCRIPTION OF THE DRAWINGS**

19 The same reference numbers are used throughout the drawings to reference
20 like components and features.

21 Fig. 1 shows a software licensing system.

22 Fig. 2 shows a block diagram of a computer used to implement the software
23 licensing system.

24 Fig. 3 shows a functional block diagram showing software components and
25 databases that implement the software licensing system.

1 Fig. 4 shows steps in a method for issuing a license pack of individual
2 licenses.

3 Fig. 5 shows steps in a method for initiating a connection between a client
4 and a server and determining whether the client has a valid license.

5 Fig. 6 shows steps in a method for distributing a software license to a client.

6 Fig. 7 shows steps in a method for challenging a client prior to granting a
7 software license to that client.

8 Fig. 8 shows steps in a method for upgrading a software license.

9
10 **DETAILED DESCRIPTION**

11 The following discussion assumes that the reader is familiar with public key
12 cryptography. For a basic introduction to cryptography, the reader is directed to a
13 text written by Bruce Schneier and entitled, "Applied Cryptography: Protocols,
14 Algorithms, and Source Code in C," published by John Wiley & Sons, copyright
15 1994 (second edition 1996), which is hereby incorporated by reference.

16 Fig. 1 shows a system 20 for licensing software. The system 20 has a
17 licensing clearinghouse 22 that creates and issues valid software licenses to one or
18 more companies, firms, agencies, or other entities, as represented by company 24.
19 The clearinghouse 22 is a separate entity from the company 24. Examples of the
20 clearinghouse include a software manufacturer, a software vendor, or a third party
21 agent that is authorized to issue software licenses on behalf of the software
22 manufacturer or vendor.

23 The company 24 contacts the clearinghouse 22 when it desires to purchase a
24 software license to run software on the company computers. The clearinghouse 22
25 has a license generator 26 that creates a "license pack" containing a set of one or

more individual software licenses. The clearinghouse 22 encrypts the license pack using the destination license server's public key and digitally signs the license pack with a digital signature unique to the clearinghouse.

The company 24 has at least one designated license server 28. The license pack is sent to the company 24 using standard communications, such as over a data communication network (e.g., Internet) or via a portable data medium (e.g., floppy diskette, CD-ROM, etc.), and installed on the license server 28.

The license server 28 is responsible for distributing the software licenses contained in the license pack to individual clients, as represented by clients 30(1)-30(6). The license server 28 verifies the license generator's digital signature on the license pack, decrypts the contents of the license pack, and stores the individual software licenses for subsequent distribution to individual clients.

The license server 28 maintains an inventory of software licenses that have been purchased from the licensing clearinghouse 22. The license server 28 monitors the software licenses that have been granted to clients. The license server 28 can distribute licenses to new clients as long as it has available non-assigned licenses. Once the supply of non-assigned licenses is exhausted, however, the license server 28 can no longer grant licenses to the clients. The only way for the license server 28 to obtain new non-assigned licenses is to purchase a license pack from the clearinghouse 22.

Because the clients might not have network connectivity to the license server 28, one or more intermediate servers, as represented by servers 32(1) and 32(2), can act as an intermediary for the clients. Each intermediate server 32 is a common server that provides conventional resources to the clients. In addition, each intermediate server 32 has network connectivity to the license server 28 to

facilitate license distribution from the license server 28 to the clients 30. The intermediate servers 32 accept software licenses issued by the license server 28; therefore, the intermediate server associations determine the scope of the license pack to a particular license server.

The clients 30 may be directly coupled to the intermediate servers 32 via a LAN (local access network) or WAN (wide area network), as represented by clients 30(1)-30(4). Additionally, the clients 30 may be indirectly coupled to the intermediate servers 32, such as using a dialup connection as represented by clients 30(5) and 30(6).

When a client 30 connects to the intermediate server 32, it must present a valid license. If the client does not have an appropriate license, the intermediate server 32 assists the client in obtaining a license from the license server 28. This provides an automated mechanism for distributing licenses to clients. The license server 28 initially checks if the requesting client already has been issued a license. When this situation is detected, the license server 28 issues the existing license to the client. This allows the client to gracefully recover licenses when they are lost.

In one particular implementation, the license server 28 determines an appropriate type of license based in part on the client's platform operating system type. The license server 28 derives the platform information by establishing a trust relationship with the client 30 and then querying its platform type. Once a client 30 has obtained a license, it is responsible for managing the storage of that license. The platform challenge process is described below in more detail.

1 **Exemplary Computer Used to Implement Servers and/or Client**

2 The license generator 26, license server 28, and intermediate server 32 are
3 preferably implemented as computer servers, such as Windows NT servers that run
4 Windows NT server operating systems from Microsoft Corporation or UNIX-
5 based servers. It is noted, however, that the license generator 26 and license server
6 28 may be implemented using other technologies, including mainframe
7 technologies, as long as they share an inter-operable communication mechanism
8 like remote procedure call (RPC) and these systems are secure.

9 The clients 30 can be implemented as many different kinds of computers,
10 including a desktop personal computer, a workstation, a laptop computer, a
11 notebook computer, a handheld PC, and so forth. The clients 30 may further
12 represent a terminal device, which is a low cost machine with limited local
13 processing and local memory. The terminal device includes a display, a keyboard,
14 a mouse (optional), limited computer resources like memory, and enough
15 intelligence to connect to an intermediate server. All applications run at the server.
16 The terminal merely provides a connection point to the server-based processing.

17 The clients 30 might also represent a network-centric computer, such as a
18 Network Computer (or NC) or a Net PC.

19 Fig. 2 shows an example implementation of a computer 40, which can be
20 used to implement the license generator 26, license server 28, and intermediate
21 server 32. The server 40 includes a processing unit 42, a system memory 44, and a
22 system bus 46 that interconnects various system components, including the system
23 memory 44 to the processing unit 42. The system bus 46 may be implemented as
24 any one of several bus structures and using any of a variety of bus architectures,
25 including a memory bus or memory controller, a peripheral bus, and a local bus.

1 The system memory 44 includes read only memory (ROM) 48 and random
2 access memory (RAM) 50. A basic input/output system 52 (BIOS) is stored in
3 ROM 48.

4 The computer 40 has one or more of the following drives: a hard disk drive
5 54 for reading from and writing to a hard disk or hard disk array, a magnetic disk
6 drive 56 for reading from or writing to a removable magnetic disk 58, and an
7 optical disk drive 60 for reading from or writing to a removable optical disk 62
8 such as a CD ROM or other optical media. The hard disk drive 54, magnetic disk
9 drive 56, and optical disk drive 60 are connected to the system bus 46 by a hard
10 disk drive interface 64, a magnetic disk drive interface 66, and an optical drive
11 interface 68, respectively. The drives and their associated computer-readable
12 media provide nonvolatile storage of computer readable instructions, data
13 structures, program modules and other data for the computer 40.

14 Although a hard disk, a removable magnetic disk 58, and a removable
15 optical disk 62 are described, other types of computer readable media can be used
16 to store data. Other such media include magnetic cassettes, flash memory cards,
17 digital video disks, Bernoulli cartridges, random access memories (RAMs), read
18 only memories (ROM), and the like. Additionally, the computer 40 may be
19 configured to serve data stored on an independent storage systems, such as disk
20 array storage systems.

21 A number of program modules may be stored on the hard disk, magnetic
22 disk 58, optical disk 62, ROM 48, or RAM 50. These programs include a server
23 operating system 70, one or more application programs 72, other program modules
24 74, and program data 76. The operating system 70 is preferably a Windows-brand
25 operating system such as Windows NT, Windows 95, Windows CE or other form

1 of Windows. The operating system 70 may alternatively be other types, including
2 Macintosh and UNIX-based operating systems.

3 A user may enter commands and information into the computer 40 through
4 input devices such as a keyboard 78 and a mouse 80. Other input devices (not
5 shown) may include a microphone, joystick, game pad, satellite dish, scanner, or
6 the like. These and other input devices are connected to the processing unit 42
7 through a serial port interface 82 that is coupled to the system bus 46, but may
8 alternatively be connected by other interfaces, such as a parallel port, game port, or
9 a universal serial bus (USB).

10 A monitor 84 or other type of display device is also connected to the system
11 bus 46 via an interface, such as a video adapter 86. The computer 40 has a
12 network interface or adapter 88, a modem 90, or other means for establishing
13 communications over a network 92.

14 15 System Architecture

16 Fig. 3 shows an exemplary software/hardware architecture of the system
17 20. The architecture includes four components: a license generator 26, a license
18 server 28, a client 30, and an intermediate server 32. The license generator 26
19 produces license packs for a fee and the license server 28 consumes the licenses by
20 installing them. In turn, the license server 28 distributes a license to the client 30
21 with the help of the intermediate server 32. The client 30 then uses the license to
22 gain access to the resources provided by the intermediate server 32.

23 The entity or organization that owns, or is responsible for, the license server
24 28 registers itself with an independent certifying authority that is trusted by both
25 the organization and the clearinghouse. The organization submits information

1 identifying itself and various license servers to the certifying authority. The
2 certifying authority performs a verification analysis of the organization to verify
3 that it is a real entity and that the identification information is true and accurate.
4 The certifying authority issues a certificate to the organization. The certificate
5 contains the public key of the organization (or particular license server), which is
6 signed by the certifying authority. This certificate becomes the license server's
7 certificate during the initial purchase request process when the license server
8 requests a license pack from the clearinghouse.

9 Similarly, the clearinghouse also registers with the certifying authority to
10 receive a public certificate. The clearinghouse certificate contains the
11 clearinghouse's public key, signed by the certifying authority.

12 The license generator 26 has a master license database 100, a licensing
13 producer 102, and a request handler 104. The request handler 104 receives a
14 purchase request 106 from the license server 28 asking to purchase one or more
15 license packs. The purchase request includes information pertaining to the licenses
16 and license server 28. For example, the purchase request might contain such
17 information as a license server ID, the license server's certificate (which contains
18 the license server's public key), a client's platform type, the quantity of licenses
19 desired, a product ID, and a list of features that the licenses should enable.
20 Additional information about a customer (e.g., name, contract number, etc.) may
21 also be requested for purposes of tracking and report generation. This information
22 is stored in the master license database 100.

23 In response to the request, the license producer 102 generates one or more
24 license packs 108, each of which contains a set of one or more non-assigned
25 licenses that are purchased from the license clearinghouse. The license generator

26 creates licensing packs in a way that prevents them from being copied and installed on multiple license servers 28 or being applied multiple times on the same server. In the preferred implementation, this is accomplished using IDs and cryptographic tools. The license producer 102 assigns a unique license pack ID to each license pack and associates the license pack ID with the license server 28 in the master license database 100. The license pack ID is embedded in the license pack 108. This prevents users from multiplying the number of licenses they purchase by installing the same license pack multiple times on the same license server.

The license generator 26 encrypts the license packs 108 with the license server's public key to ensure protected transport to the license server 28 and to ensure that only the license server 28 can open the packs 108. The license generator 26 also digitally signs the license packs 108 with a private signing key of the license generator 26. The license server 28 uses this signature to validate that the license pack came from an authorized license generator and has not been altered.

The license pack 108 is a data structure that contains various information to enable the license server to distribute software licenses. The data structure contains fields with the licensing information. Table 1 shows the data fields of a license pack data structure.

Table 1: License Pack Contents

<u>Field</u>	<u>Description / Purpose</u>
Message Version	An ID used to distinguish different versions of the data structure.
License Pack Serial	A serial number assigned by the license

1 Manufacturer-Specific
2 Product Data

Manufacturer-dependent information used
to identify the product. As an example,
this data might include:

- 3 1. Product Family Code
- 4 2. Product Version
- 5 3. License Type

6 Signature

Digital signature generated by the license
generator using the clearinghouse private
key.

8
9 Clearinghouse's Public
10 Key Certificate

The certificate issued to the clearinghouse
and containing the clearinghouse's public
key. This public key is used to sign the
encrypted license pack.

11
12
13
14 One parameter of the purchase request and subsequent license pack is the
15 client platform type. As one possible implementation, the system 20 is configured
16 to reliably recognize four different platform types: Windows, Non-Windows,
17 Legacy, and Direct-Connect. A "Windows"-type platform means the client
18 computer runs a 32-bit version of Microsoft Windows operating system (e.g.,
19 Windows 95, Windows 98, Windows NT, etc.). A "Non-Windows"-type platform
20 means the client computer runs an operating system other than a Windows brand
21 operating system. A "Legacy"-type platform indicates that the client runs an older
22 version of an operating system that cannot be adequately determined by the license
23 server as a "Windows"-type or a "Non-Windows"-type. A "Direct-Connect"
24 platform means the client is a terminal that attaches directly to the server's bus and
25

thus, all of the operating system functionality is provided directly by the server.

Table 2 summarizes the platform types.

Table 2: Platform Types

<u>Platform Type</u>	<u>Description</u>
Windows	Authenticated client platforms that are Win32-based.
Non-Windows	Authenticated client platforms that are not Win32-based.
Legacy	Clients that are implemented with older operating systems that are incapable of fielding a client platform challenge from the license server. There is no way of determining whether or not the client's platform is Win32 capable.
Direct-Connect	Multi-console clients that are attached directly to the server's BUS. These clients derive the operating system capabilities from the server itself.

The license server 28 has a license pack installer 110 and a secure license store 112. The license pack installer 110 installs the license pack(s) 108 received from the license generator on the secure license store 112. The license pack installer 110 may also be used to order the license packs, when such purchase requests are made electronically.

The license pack is stored in a secured database. A library of routines for adding, removing, querying, upgrading and extracting licenses are used to manage the licenses within the license store. As noted above, the license packs are encrypted using the license server's private key to prevent users from tampering

1 with the licenses or moving them to another license server. License store APIs
2 (application program interfaces) are used to encrypt the licenses as they are placed
3 on the secure license store 112 and to decrypt the licenses as they are removed
4 from the store.

5 To prevent the same licenses from being applied multiple times on the same
6 license server, each license pack 108 contains a unique license pack ID assigned
7 by the license generator 26 when the license pack is created. The licenses are
8 stored in the license store 112 based on the license pack ID.

9 The license store 112 contains two tables: a license pack (LP) table 114 and
10 a client assignment (CA) table 116. The license pack table 114 records
11 information pertaining to the license packs 108. The license pack table 114 is
12 indexed using the license pack ID, which enables quick access and a convenient
13 way to check if a particular license pack is already installed in the secure store.

14 Table 3 shows the fields in the license pack table 114.
15
16
17
18
19
20
21
22
23
24
25

Table 3: License Pack Table

<u>Field</u>	<u>Description</u>
License Pack ID	A unique identifier assigned by the license generator.
Quantity	The number of software licenses contained in the license pack.
Number Assigned	The number of software licenses that have been assigned to clients.
First Active Date	The date on which the licenses within the license pack can first be used.
Expiration Date	The date on which the software licenses in the license pack will expire.
Begin Serial Number	The beginning serial number for the licenses in the license pack. The number is used to assign a unique serial number to each license within the license pack.
Product-Specific Attributes	Product-dependent information to indicate specific features of a product. As an example, this data might include: <ul style="list-style-type: none"> 1. Product ID 2. Product Flags 3. Platform Type

The number assigned field need not be kept, but it helps eliminate the need to count the number of assigned licenses each time an administrator wants to determine how many free licenses are available.

The client assignment table 116 contains a list of all licenses that have be distributed to the clients. Each record in the client assignment table 116 is

assigned a unique license ID. The license ID serves two purposes: (1) it allows the table 116 to be indexed and (2) it provides a license tracking mechanism for the client. The client assignment table 116 also contains the license pack ID from which each license is derived.

Table 4 shows the fields in the client assignment table 116.

Table 4: Client Assignment Table

<u>Field</u>	<u>Description</u>
License ID	A unique identifier assigned by the license server to each software license, based on the begin serial number.
License Pack ID	The unique identifier assigned by the license generator.
Client ID	A unique identifier of the client to which the software license is granted.
Issue Date	The date on which the software license is issued to the client.

The license pack ID fields in the license pack table 114 and the client assignment table 116 can be used to join the tables in a one-to-many relationship; that is, one record identified in the license pack table 114 to many records in the client assignment table 116 as software licenses are issued to clients. This joiner yields a list of all software licenses assigned to clients from a given license pack. The client ID field enables the administrator to query all licenses for a particular client.

license server, and uses this public key to verify the client's signature before installing the client image on the cache 120.

The license server 28 also has a request handler 122, a client authenticating module 124, and a granting module 126. The request handler 122 receives requests for software licenses from clients. The client request typically includes the client ID. The request handler 122 passes the request to the client authenticating module 124, which determines whether the client is authentic and able to receive a software license.

As part of the authentication process, the client authenticating module 124 initiates a platform challenge requesting a client executable image from the client 30. One preferred approach to performing a platform challenge is described below in more detail under the sub-heading “Platform Challenge”.

The client authenticating module 124 compares the client executable image received from the client to the client executable image stored in the client image cache 120. The client is deemed authentic if the two images match. The client authenticating module 124 informs the granting module 126 when the client is authenticated.

The granting module 126 grants a software license from the secure license store 112 to the authenticated client. To prevent an issued license from being copied from machine to machine, the software license is assigned to a specific client by assigning a client ID to the license and including that ID within the license. The software license is also given a license ID. The license ID is associated with the client ID in the client assignment table 116 to track which client receives the issued license.

The license server 28, based on information derived from the license pack, fills in fields of a license data structure at the time the license is issued. As one example, the license data structure is implemented using an X.509 certificate, which is well known in the art. The license server 28 then digitally signs the software license using a signing key that is not disclosed to the client. Table 5 shows the data fields of a software license data structure.

Table 5: Software License Contents

<u>Field</u>	<u>Description / Purpose</u>
Version	Identifies the "data structure" version of the software license so newer licenses can be used on older servers.
License ID	A unique ID assigned to the software license by the license server at the time of issuance to the client.
Client ID	The unique identifier of the client to which the software license is assigned.
Issue Date	The date on which the software license is assigned to the client.
Expiration Date	The date on which the software licenses in the license pack will expire.
Product-Specific Attributes	Product-dependent information to indicate specific features of a product. As an example, this date might include: <ul style="list-style-type: none"> 1. Product ID 2. Product Flags 3. Platform Type
Signature	Digital signature generated by the license generator using the clearinghouse private key.

1 License Server's The license server's public key in certificate form,
2 Certificate as issued by the certifying authority.
3
4

5 As part of the granting process, the client assignment table 116 is updated to
6 reflect that a particular license having a specific license ID is issued to a particular
7 client having a specific client ID. Additionally, the number assigned field in the
8 license pack table 114 is updated to reflect that another license has been assigned
9 to a client.

10 The license pack installer 110, client image installer 118, request handler
11 122, client authenticating module 124, and granting module 126 are preferably
12 implemented as software programs executing on the license server 28. These
13 software programs are preferably implemented as part of the operating system at
14 the license server.

15 The intermediate server 32 acts as a go between for the client 30 and license
16 server 28. The intermediate server is a full-service server that is used regularly by
17 the client to perform normal tasks that are customary for the company or entity.
18 But, the intermediate server is further equipped with a client licensing unit 128 to
19 facilitate communication between the client 30 and license server 28. The
20 intermediate server 32 also has a legacy license store 130, which stores licenses for
21 clients whose platforms cannot generate a unique system ID.

22 The client 30 has a license requestor 132, a challenge handler 134, and a
23 license cache 136. The license requestor 132 initiates the license requests for
24 obtaining a software license from the license server 28. This involves connecting
25 to the intermediate server 32 and presenting a software license and a client ID to

1 the intermediate server 32. The client ID submitted by the client is validated
2 against the client ID within the license. To prevent a client from simply looking
3 within a license to find its associated client ID, the license server encrypts the
4 software license with a key that is never disclosed to clients and hence the client is
5 incapable of decrypting the software license. Furthermore, license tampering is
6 prevented by digitally signing the software licenses when the license server issues
7 them.

8 The client ID is passed onto the license server 28, which then initiates a
9 platform challenge. The client's challenge handler 134 handles the platform
10 challenge from the license server 28. It computes a response to the challenge that
11 contains the client's image, which can be used by the license server 28 to
12 authenticate the client.

13 If the client is deemed authentic, the license server downloads a software
14 license to the client. The license server 28 encrypts the license using the client's
15 public key and digitally signs the license. Additionally, the license generator
16 assigns a unique license ID to the issued license. Because the licenses are tied to a
17 specific client through a client ID, digitally signed by the license server and
18 encrypted, the software licenses cannot be activated on other clients.

19 The license requestor 132 verifies the signature on the license to confirm
20 that it came from the license server 28 and stores the software license in the license
21 cache 136. It is the responsibility of the license requestor 132 to manage the
22 licenses stored in the cache 136. The licenses are organized in the license cache
23 136 according to information about the license issuing authority and product ID.

24 The license cache 136 is kept in persistent (non-volatile) storage. Clients
25 that do not have persistent storage can be issued licenses as long as they can

1 generate a unique client ID and can respond to the client platform challenge
2 protocol. The licensing system handles this case in the same way it recovers lost
3 licenses. On connect, the intermediate server contacts the license server for a new
4 license. The license server realizes, through the system ID, that the license has
5 already been issued. In this case, the old license is simply returned to the client.
6 Clients that cannot generate a system ID or respond to the platform challenge
7 protocol use the legacy licenses stored in the legacy license store 130 at the
8 intermediate server 32.

9 The license requestor 132 and the challenge handler 134 are preferably
10 implemented in software executing on the client 30. These software programs are
11 preferably implemented as part of the client's operating system.

12 It is noted that Fig. 3 illustrates one possible implementation of the software
13 licensing system 20. Other implementations are possible. As one example, the
14 components associated with a client platform challenge may be removed. These
15 components include the client image installer 118, the client image cache 120, and
16 the client authenticating module 124 in the license server 28, and the challenge
17 handler 134 in the client 30.

18 19 System IDs

20 One aspect of system 20 is the ability to generate unique identifiers for the
21 servers and clients. These unique IDs include the license server ID in license
22 server certificate 140 and the client's system ID 142 (collectively referred to as
23 "System IDs"). The system 20 employs a per-seat licensing technique, in which
24 licenses are associated with a particular client or machine (i.e., "seat" or "node").
25 The license server certificate 140 contains a unique ID for the license server 28,

1 which is passed to the license generator during a request for a license pack. The
2 client's system ID 142 is a unique identifier of the client computer. It is noted that
3 the client ID assigned by the license server to a software license may be client's
4 system ID, although it will typically be a separate identifier created by the license
5 server solely for tracking purposes.

6 As one possible implementation, the system IDs can be based on
7 information collected from a computer's hardware and installed software. For
8 example, hard disk volume numbers, network cards, registered software, video
9 cards, and some microprocessors contain unique identifiers. On PCs, this
10 information can be combined to uniquely identify a particular PC. Other
11 information that might be used includes total RAM and floppy disk drive
12 configuration. Because these components can be removed or replaced, thus
13 changing the system ID, procedures for accepting system IDs allow for some
14 variations. For instance, the procedures might allow for a few parameters to vary.

15 However, relying on a machine's hardware characteristics may not always
16 be sufficient when generating unique machine IDs. For example, the hardware
17 characteristics of some computers may not vary, so they would all generate the
18 same machine ID. In these cases, manufacturers "brand" the computers with a
19 unique identifier that it can be used to generate a unique machine ID. Client
20 platforms that cannot generate a unique machine ID are still permitted to connect
21 to an intermediate server and are deemed legacy platforms. Legacy licenses
22 maintained in the legacy license store 130 are used for these machines.
23
24
25

Issuance of License Pack

Fig. 4 shows steps in a method for requesting and issuing a license pack from a license generator. At step 150, the license server 28 generates and sends a purchase request 106 to an authorized license generator 26. The request 106 contains information used by the license generator 26 to issue one or more software license packs to the requesting license server 28. The purchase request 106 contains the platform type (see Table 2), the quantity of licenses desired, the product ID, the license server's certificate (containing the license server's public key K_{LS_pub} and the license server ID), and the list of features that the license should enable. The license server can submit this information electronically to the license generator via the Internet, modem, e-mail, on a floppy diskette, or other electronic means. Additionally, the administrator at the company or entity might submit a purchase request to the licensing clearinghouse 22 in writing on paper, or place an order orally by telephone. The license server 28 typically submits a licensing fee with the purchase request, or sometime following the initial communication.

After collecting the fee for the software licenses, the license generator 26 creates a license pack containing a set of one or more individual software licenses and assigns a unique license pack ID to the license pack (step 152 in Fig. 4). The license generator 26 stores the collected information in the master license database 100 (step 154). The information from the license server 28 is correlated within the database 100 to the license pack ID. In this manner, the license pack ID is associated with a particular license server having a specific license server ID (step 156).

1 The license generator 26 encrypts the license pack of software licenses
2 using the license server's public key K_{LS_pub} , thus binding the license pack to the
3 requesting license server 28. The license generator 26 digitally signs the license
4 pack using its (i.e., the clearinghouse's) private signing key K_{CH_pri} (step 160 in
5 Fig. 4) and sends the license pack to the requesting license server 28.

6 The license pack 108 contains a set of one or more non-assigned licenses
7 and the license pack ID. Table 1 lists the contents of the license pack 108.

8 At step 164 in Fig. 4, the license server 28 uses the clearinghouse's public
9 signing key K_{CH_pub} to verify that the digital signature accompanying the license
10 pack 108 belongs to the license generator 26 of clearinghouse 22 and that the
11 license pack 108 has not been altered. If the signature is authentic and from a
12 known clearinghouse, the license server 28 decrypts the license pack contents
13 using its private key K_{LS_pri} (step 166). The license server 28 extracts the license
14 pack ID and queries the secure license store 112 to see if it already contains the
15 same license pack (step 168). If the license pack is new, the license server installs
16 it on the secure license store 112 (step 170).

17 18 Distribution of Licenses

19 Client Connection

20 Fig. 5 shows steps in a process that facilitates a client's initial connection to
21 the intermediate server. The client connects to the intermediate server 32 to ask
22 for services or data provided by the server. Prior to working with the client and
23 providing access to files, the intermediate server 32 wants to verify first that the
24 client has a valid software license issued by a recognized license server. The client
25 30 may or may not have a valid license, so the intermediate server makes an initial

1 evaluation when the client attempts to connect. Generally, if the client 30 has a
2 valid license, the client is permitted to connect and use the server's resources. If
3 the client 30 offers an invalid license, the client is disconnected. If the client 30
4 does not offer a valid license or offers an expired license, the intermediate server
5 32 facilitates the process of obtaining a new software license.

6 At step 172, the client 30 submits a connection request to the intermediate
7 server 32. The connection request includes the client's system ID that uniquely
8 identifies the computer. In response, the intermediate server 32 passes a list of the
9 product IDs required (step 174). In this manner, the intermediate server 32 limits
10 its acceptance of software licenses to those that are issued by legitimate and
11 authorized license servers.

12 With this information, the client 30 queries its license cache 136 to search
13 for a suitable license from a license server that appears on the list (step 176 in Fig.
14 5). If a software license is found, the client 30 sends the software license to the
15 intermediate server 32 along with the client ID; otherwise, the client 30 submits
16 only a client ID (step 178). The software license contains the digital signature of
17 the license server.

18 At step 180 in Fig. 5, the intermediate server 32 determines whether the
19 client submitted a software license. If so, the intermediate server 32 verifies
20 whether the digital signature belongs to an authorized license server and whether
21 the license contains a valid client ID (step 182). The client ID is checked by
22 extracting the client ID from the license (which was provided originally by the
23 licensing server, as described below) and comparing it to the client ID received
24 from the client. If the two match, the client ID passes.
25

1 If the digital signature or the client ID is not valid (i.e., the “not valid”
2 branch from step 182), the software license is deemed invalid. The client’s request
3 for connection is then rejected and the client is disconnected. On the other hand, if
4 the digital signature and the client ID are both valid (i.e., the “valid” branch from
5 step 182), the intermediate server 32 checks if the license has expired (step 184),
6 the connection is completed if the license is still valid i.e. has not expired and the
7 client is allowed access to the services and files of the intermediate server (step
8 186).

9 In the event that the client 30 does not submit a valid license or submits an
10 expired license, the intermediate server requests a new software license from the
11 license server (step 188 in Fig. 5).

12 13 New License Grant

14 Software licenses are distributed to the client automatically by the license
15 server. As discussed above, when a client 30 connects to an intermediate server
16 32, the client must present a valid license. If it cannot, the intermediate server acts
17 as a proxy for the client and requests a license from its associated license server.

18 Fig. 6 shows steps in a method for granting a new software license from the
19 license server 28 to the client 30. The method begins with step 188, which is the
20 same new license request discussed above with respect to step 188 of Fig. 5. The
21 new license request includes the client’s system ID and the product ID. In
22 response to the request, the license server 28 initiates a client challenge to
23 determine who the client is and what platform it is running (step 190). In general,
24 this involves generating a challenge and sending it to the intermediate server 32
25

1 (step 192). The intermediate server 32 forwards the challenge to the client 30
2 (step 194).

3 At step 196 in Fig. 6, the client responds to the challenge in a manner that
4 provides trusted information about client, including the platform type and the
5 client's public key. The response is passed to the intermediate server 32, which
6 forwards it to the license server 28 (step 198).

7 At step 200 in Fig. 6, the license server determines whether the response is
8 proper, and hence, whether the client is authentic. If the client is authenticated
9 (i.e., the "yes" branch from step 200), the license server proceeds with granting a
10 software license. The license server 28 first queries the secure license store 112 to
11 determine if a license for that client has already been issued (step 202). This
12 procedure accommodates the case in which the client has lost its valid software
13 license. If a non-expired license is found, the license server 28 forwards it to the
14 client 30.

15 Otherwise, the license server 28 attempts to allocate a software license for
16 the client, assuming a non-assigned license still exists in the license pack. If a
17 license can be allocated, the license server 28 retrieves a software license that is
18 appropriate for the client's platform from the secure software store 112 and grants
19 the software license to the client (step 204 in Fig. 6). The license server 28 adds a
20 record to the client assignment table 116 and the corresponding number assigned
21 field is updated to reflect one additional allocation.

22 To prevent the software license from being copied from one client machine
23 to another, the software license is assigned to the specific client by including its
24 client ID within the license. The software license also has a corresponding license
25 ID that is associated with the client ID in a the client assignment table 116 in the

1 secure license store 112 at the license server. The contents of the license are
2 described above in Table 5.

3 The license server 32 digitally signs the software license (step 206) and
4 encrypts it using the client's public key K_{C_pub} (step 208), thereby binding the
5 license to the client. The encrypted license is forwarded to the intermediate server
6 32, which passes it on to the client 30 and completes the connection (step 210). By
7 encrypting the license, the client or the license server need not trust the
8 intermediate server because the intermediate server cannot maliciously utilize or
9 modify the encrypted license. It also removes the risk of a rogue server
10 masquerading as intermediate server. At step 212, the client 30 decrypts the license
11 using the client's private key K_{C_pri} and stores the license in the license cache 136.

12 In the event that the client's response to the challenge is deemed improper
13 (i.e., the "no" branch from step 200), the license server returns a rejection notice
14 (step 214 in Fig. 6). This rejection notice is passed on by the intermediate server
15 32 (step 216) and used to inform the user (step 218).

16 17 Platform Challenge

18 Fig. 7 shows a more detailed method for providing a platform challenge to
19 the client. In this illustration, the intermediate server 32 is shown as the go
20 between, with the forwarding steps omitted for ease of description.

21 An aspect of platform validation is establishing the authenticity of the
22 client. The system utilizes the client's executable image to generate a digital
23 signature that uniquely identifies the client. As noted above, the client's
24 executable image is available to the license server 28 because it is stored in the
25 client image cache 120.

1 When a client requests a software license from the license server, the client
2 30 submits a client software ID (step 220 in Fig. 7). The software ID is assigned
3 by the software manufacturer/vendor to be unique for each client release. The
4 client software ID is a bit field that contains a platform identifier, a vendor
5 identifier, and a client revision field. The arrangement of the bits depends on how
6 many platforms and clients are supported.

7 At step 222, the license server 28 uses the software ID to lookup the client's
8 executable image in the client image cache 120. If the image is not present in the
9 cache (i.e., the "no" branch from step 222), the client is denied a software license
10 and a rejection is returned to the client and informs the user (steps 224 and 226).

11 On the other hand, if an image is present (i.e., the "yes" branch from step
12 222), the license server 28 sends a challenge to the client 30 to establish a trust
13 relationship with the client (step 228). The challenge is preferably a 128-bit
14 random number.

15 The client 30 applies a one-way function to a combination of the challenge
16 and the client's image (step 230). Preferably, the client concatenates the challenge
17 and the client image and computes a hash value, as follows:

$$\text{Challenge Response} = \text{Hash}(\text{challenge}|\text{client image}|\text{challenge})$$

18
19
20
21 The client 30 sends the challenge response (i.e., the hash value) back to the
22 license server 28 (step 232).

23 Meanwhile, the license server 28 uses the software ID to retrieve a
24 reference copy of the client image from its cache 120 (step 234 in Fig. 7). The
25 license server then computes a test hash value using the same hash function, and a

1 concatenated version of the same 128-bit challenge and the client image retrieved
2 from the cache 120 (step 236).

3 The license server 28 compares the test hash value (H') with the hash value
4 (H) returned from the client (step 238). If the two values are the same, the client's
5 platform information is extracted from the client software ID and a trust
6 relationship established (i.e., the "yes" branch from step 238). Otherwise, the
7 client is denied a software license and a rejection is returned to the client (i.e., the
8 "no" branch from step 238).

9 10 Upgrading Licenses

11 The process for upgrading an existing license is very similar to the license
12 distribution process. The primary difference is that a platform challenge is not
13 performed because a valid, digitally signed license is presented to the license
14 server.

15 Fig. 8 shows the steps in a method for upgrading an existing license. Steps
16 172-176 are identical to those defined above with respect to Fig. 5. At step 240,
17 the client 30 submits a valid software license to the intermediate server 32.

18 At step 242 in Fig. 8, the intermediate server 32 determines whether the
19 license has expired and/or is for an older version. Assuming it meets one of these
20 conditions, the intermediate server automatically contacts the license server 28 and
21 requests that the license be upgraded (step 244). The intermediate server passes
22 the old license and the client's system ID to the license server 28.

23 The license server 28 validates the old license and extracts the license's ID,
24 which is used as an index into the client assignment table 116 in the secure license
25 store 112. The license server 28 examines the table 116 to determine whether an

1 upgrade is available (step 246). If so, the license server 28 upgrades a record in
2 the table, consuming one upgrade license, and returns an upgraded license to the
3 intermediate server 32 (step 248). The intermediate server 32 forwards the
4 upgraded license to the client and completes the connection (step 250). The client
5 30 replaces the old license with the upgraded one in the license cache 136 (step
6 252).

7 As a matter of policy, licenses are assumed to be backward compatible.
8 That is, a next generation 5.X license is always accepted by a current generation
9 4.X server. This allows a customer to have a seamless mix of different servers.
10 Variances in the licenses internal data structures are taken into account by
11 including a version number within the license.

12 Temporary Licenses

13
14 Suppose a client 30 requests a software license, but the license server 28
15 does not have an available license in the secure license store. In this case, the
16 license server 28 issues a temporary license that is valid for a finite duration (e.g.,
17 60 days).

18 With reference to Fig. 3, the requesting client submits its system ID 142 to
19 the intermediate server 32, which forwards the client's system ID 142 to the
20 license server 28. The license server 28 generates a temporary license and
21 associates it with the client's system ID 142. The temporary license is passed back
22 through the intermediate server 32 to the client 30. Each time the client presents
23 the temporary license, a new license request is generated. Once the license server
24 has an available license (e.g., the license server purchased additional licenses from
25

1 the license clearinghouse), it issues a permanent license to the client. Temporary
2 licenses are replaced only by a valid permanent license.

3 When a temporary license expires, the license server 28 no longer accepts it
4 and services are denied. Furthermore, the client is only granted one temporary
5 license and will not be permitted to request a second temporary. If a client
6 attempts to request a second temporary license, the license server will detect the
7 system ID and recognize that this ID is already associated with a previously issued
8 temporary license. The license server 28 simply returns the previously issued
9 temporary license, which is inoperable because it has expired.

10 Although the invention has been described in language specific to structural
11 features and/or methodological steps, it is to be understood that the invention
12 defined in the appended claims is not necessarily limited to the specific features or
13 steps described. Rather, the specific features and steps are disclosed as preferred
14 forms of implementing the claimed invention.

1 **CLAIMS**

2 1. A computer-implemented method comprising the following steps:
3 creating a license pack at a license generator, the license pack containing a
4 set of one or more individual software licenses;
5 signing the license pack with a digital signature of the license generator;
6 issuing the license pack to a license server;
7 verifying, at the license server, the license generator's digital signature on
8 the license pack; and
9 distributing the software licenses contained in the license pack from the
10 license server to corresponding clients.

11
12 2. A computer-implemented method as recited in claim 1, further
13 comprising the step of creating a license pack containing a predefined number of
14 software licenses.

15
16 3. A computer-implemented method as recited in claim 1, further
17 comprising the following steps:

18 creating a license pack ID at the license generator; and
19 evaluating the license pack ID at the license server.

20
21 4. A computer-implemented method as recited in claim 1, further
22 comprising the following steps:

23 encrypting the license pack at the license generator; and
24 decrypting the license pack at the license server.
25

1 5. A computer-implemented method as recited in claim 1, further
2 comprising the step of creating a license pack that is tailored to a particular
3 operating platform of the clients.
4

5 6. A computer-implemented method as recited in claim 1, further
6 comprising the step of determining an authenticity of an individual client prior to
7 distributing the software license to that individual client.
8

9 7. A computer-implemented method as recited in claim 1, further
10 comprising the following steps:

11 determining whether an individual client has a non-expired license;

12 in the event that the client has a non-expired license, forwarding the non-
13 expired license to the individual client; and

14 in the event that the client does not have a non-expired license, granting one
15 of the software licenses from the license pack.
16

17 8. A computer-implemented method as recited in claim 7, further
18 comprising the step of encrypting said one software license using a public key of
19 the individual client.
20

21 9. A computer-implemented method as recited in claim 1, further
22 comprising the step of evaluating whether an individual client already has a
23 software license that has not yet expired.
24
25

1 10. A computer-implemented method as recited in claim 1, wherein the
2 license pack has a license pack ID, and further comprising the step of granting
3 additional licenses for the license pack having the same license pack ID.
4

5 11. A computer-implemented method as recited in claim 1, further
6 comprising the following steps:

7 submitting an old software license from one of the clients to the license
8 server, the old software license containing a license ID;

9 determining whether an upgrade software license is available for the license
10 ID;

11 granting the upgrade software license if available to the one client; and

12 replacing, at the one client, the old software license with the upgrade
13 software license.
14

15 12. A computer-implemented method as recited in claim 1, further
16 comprising the step of distributing a temporary license of in an event that no more
17 licenses are available from the license pack.
18

19 13. A computer-implemented method for distributing software licenses
20 to clients so that the clients may legally execute underlying software to which the
21 software licenses pertain, the computer-implemented method comprising the step
22 of electronically issuing the software licenses as digital certificates that can be
23 distributed in one-to-one correlation with individual clients and traced to an
24 issuing authority.
25

1 14. A computer-readable medium having computer readable instructions
2 for performing the step as recited in claim 13.

3
4 15. A computer-implemented method comprising the following steps:
5 receiving a request for a software license from a particular license server;
6 creating a license pack containing a set of one or more individual software
7 licenses;
8 assigning a license pack ID to the license pack;
9 associating the license pack ID with the particular license server;
10 digitally signing the license pack; and
11 issuing the signed license pack to the particular license server.

12
13 16. A computer-implemented method as recited in claim 15, further
14 comprising the step of creating a license pack containing a predefined number of
15 software licenses.

16
17 17. A computer-implemented method as recited in claim 15, further
18 comprising the step of creating a license pack that includes a platform type
19 indicating a type of operating platform for which the software licenses can be used.
20
21
22
23
24
25

1 18. A computer-implemented method as recited in claim 15, further
2 comprising the step of creating a license pack that includes a predefined number of
3 software licenses, a platform type indicating a type of operating platform for which
4 the software licenses can be used, an expiration date indicating a date on which the
5 software licenses will expire, and a product ID that identifies a product with which
6 the software licenses can be used.

7
8 19. A computer-implemented method as recited in claim 15, further
9 comprising the step of encrypting the license pack.

10
11 20. A computer-readable medium having computer readable instructions
12 for performing the steps as recited in claim 15.

13
14 21. A computer-implemented method comprising the following steps:
15 receiving a request for a software license from a particular client;
16 determining an authenticity of the particular client;
17 selecting a software license from a pack of software licenses that is
18 appropriate for the particular client, the software license having an associated
19 license ID;
20 associating the license ID with the particular client; and
21 granting the software license to the particular client.

1 22. A computer-implemented method as recited in claim 21, further
2 comprising the step of determining whether the particular client already has a non-
3 expired license, and if so, forwarding the non-expired license to the particular
4 client rather than granting the software license.

5
6 23. A computer-implemented method as recited in claim 21, further
7 comprising the step of granting the software license as containing the license ID, a
8 platform type indicating a type the platform, an issue date indicates a date on
9 which the license is issued to the client, an expiration date that indicates a date on
10 which the software license will expire, a product ID that identifies a product with
11 which the software licenses can be used, a client ID that identifies the particular
12 client, and a version of the software license.

13
14 24. A computer-implemented method as recited in claim 21, wherein the
15 step of determining the authenticity comprises the following steps:
16 receiving a client software ID from the particular client; and
17 evaluating the client software ID to determine whether the client is
18 authentic.

19
20 25. A computer-implemented method as recited in claim 21, wherein the
21 step of determining the authenticity comprises the following steps:
22 maintaining a set of client images;
23 receiving a client software ID from the particular client; and
24 comparing the client software ID to the client images to evaluate whether
25 the client is authentic.

1
2 26. A computer-implemented method as recited in claim 21, further
3 comprising the following steps:

4 determining a platform of the particular client; and
5 selecting the software license as is appropriate for the platform of the
6 particular client.

7
8 27. A computer-implemented method as recited in claim 21, further
9 comprising the step of encrypting the software license using a public key of the
10 particular client.

11
12 28. A computer-implemented method as recited in claim 21, further
13 comprising the step of granting a temporary software license that expires in a
14 substantially shorter duration in comparison to the software license.

15
16 29. A computer-readable medium having computer readable instructions
17 for performing the steps as recited in claim 21.

18
19 30. A computer-implemented method comprising the following steps:
20 computing a value as a one-way function of a client executable image that
21 uniquely identifies a client; and
22 digitally signing the value using a private signing key of a server that serves
23 the client to create a client image digital signature that is unique to the client.
24
25

1 31. A computer-implemented method as recited in claim 30, wherein the
2 computing step comprises the step of hashing the executable image to produce a
3 hash value.

4
5 32. A computer-implemented method as recited in claim 30, further
6 comprising the step of storing the client image digital signature at the client.

7
8 33. A computer-implemented method as recited in claim 30, further
9 comprising the following steps:

10 storing the client executable image at the server;
11 storing the client image digital signature at the client;
12 submitting the client image digital signature from the client to the server
13 when requesting a software license; and
14 evaluating an authenticity of the client based on the client image digital
15 signature prior to granting a software license to the client

16
17 34. A computer-implemented method as recited in claim 33, wherein the
18 evaluating step comprises the following steps:

19 unsigned the client image digital signature using a public key of the server
20 to recover the client executable image; and

21 comparing the recovered client executable image to the client executable
22 image stored at the server.
23
24
25

1 35. A computer-implemented method as recited in claim 34, further
2 comprising the step of rejecting the request for a software license in an event that
3 the recovered client executable image does not match the client executable image
4 stored at the server.'

5
6 36. A computer-readable medium having computer readable instructions
7 for performing the steps as recited in claim 30.

8
9 37. A computer-implemented method comprising the following steps:
10 receiving a request for a software license from a client having a valid client
11 image;
12 submitting a challenge to the client, the challenge comprising a random
13 number;
14 computing, at the client, a function of the challenge and the client image to
15 produce a response;
16 returning the response to the server;
17 deriving the client image from the response at the server; and
18 verifying the client image at the server prior to granting the software
19 license.

20
21 38. A computer-implemented method as recited in claim 37, wherein the
22 computing step comprises the following steps:
23 concatenating the random number and the client image to produce a
24 concatenated value; and
25 computing a hash function of the concatenated value.

1
2 39. A computer-implemented method as recited in claim 37, further
3 comprising the step of rejecting the request for the software license in the event
4 that the client image cannot be verified.
5

6 40. A computer-implemented method as recited in claim 37, further
7 comprising the step of establishing a trust with the client and subsequently
8 granting the software license in the event that the client image can be verified.
9

10 41. Computer-readable media distributed at the server and the client
11 having computer readable instructions for performing the steps as recited in claim
12 37.
13

14 42. A computer-implemented method comprising the following steps:
15 submitting an old software license from a client to a server, the old software
16 license containing a license ID;
17 determining whether an upgrade software license is available for the license
18 ID;
19 granting the upgrade software license, if available, to the client; and
20 replacing, at the client, the old software license with the upgrade software
21 license.
22
23
24
25

1 43. A computer-implemented method as recited in claim 42, further
2 comprising the step of tracking at the server that the upgrade software license is
3 granted to the client.
4

5 44. Computer-readable media distributed at the server and the client
6 having computer readable instructions for performing the steps as recited in claim
7 42.
8

9 45. A system for licensing software, comprising:
10 a license generator to create a license pack containing a set of one or more
11 individual software licenses, the license generator digitally signing the license
12 pack with a digital signature; and
13 a license server remote from, but operatively coupled to, the license
14 generator to receive the license pack from the license generator, the license server
15 verifying the license generator's digital signature on the license pack and storing
16 the individual licenses for subsequent distribution to individual clients.
17

18 46. A system as recited in claim 45, wherein the license generator
19 assigns a license pack ID to the license pack and associates the license pack ID
20 with the license server.
21

22 47. A system as recited in claim 45, wherein the license generator
23 encrypts the license pack using a public key of the license server.
24
25

1 48. A system as recited in claim 45, wherein the license pack contains a
2 preset number of software licenses.

3
4 49. A system as recited in claim 45, wherein the license pack identifies a
5 type of operating platform for which the software licenses can be used.

6
7 50. A system as recited in claim 45, wherein the license pack comprises
8 at least one of the following items:

9 a predefined number of software licenses;

10 a platform type indicating a type of operating platform for which the
11 software licenses can be used

12 an expiration date that indicates a date on which the software licenses will
13 expire; and

14 a product ID that identifies a product with which the software licenses can
15 be used.

16
17 51. A system as recited in claim 45, wherein the license server selects a
18 software license from the license pack and grants the software license to a client,
19 the software license having a license ID and the license server associating the
20 license ID with the client.

21
22 52. A system as recited in claim 45, wherein the license server
23 challenges an authenticity of a client prior to granting a software license from the
24 license pack to the client.

1 53. A system as recited in claim 45, wherein the license server grants a
2 software license server to a particular client, the license server encrypting the
3 software license using a public key of the particular client.

4
5 54. A system as recited in claim 45, wherein the license server
6 distributes the software licenses to the individual clients via one or more
7 intermediate servers.

8
9 55. A license generator for issuing packs of software licenses to
10 authorized license servers, comprising:

11 a request handler to receive a request from a license server for a license
12 pack;

13 a license producer responsive to the request received by the request handler
14 to generate a license pack containing a set of one or more individual software
15 licenses; and

16 the license producer assigning a license pack ID to the license pack,
17 associating the license pack ID with the license server, and digitally signing the
18 license pack.

19
20 56. A license generator as recited in claim 55, wherein the license pack
21 contains a predefined number of software licenses, a platform type indicating a
22 type of operating platform for which the software licenses can be used, an
23 expiration date that indicates a date on which the software licenses will expire, and
24 a product ID that identifies a product with which the software licenses can be used.

1 57. A license generator as recited in claim 55, wherein the license
2 producer encrypts the license pack using a public key of the license server.

3
4 58. A license generator as recited in claim 55, further comprising a
5 master license database, the license producer storing the license pack ID in
6 correlation with an ID of the license server in the master license database.

7
8 59. A program embodied on a computer-readable medium, comprising:
9 a code segment to create a license pack containing a set of one or more
10 individual software licenses;
11 a code segment to assign a license pack ID to the license pack;
12 a code segment to associate the license pack ID with the particular license
13 server; and
14 a code segment to digitally sign the license pack.

15
16 60. A license server for issuing individual software licenses from a
17 software pack received from a licensing clearinghouse, comprising:
18 a license store to store the software pack of individual software licenses,
19 each software license having an associated license ID;
20 a request handler to receive a request for a software license from a client;
21 a client authenticating module to determine whether the client is authentic
22 and can receive a software license; and
23 a granting module to grant a software license from the license store to an
24 authenticated client and to associate the license ID with the authenticated client.
25

1 61. A license server as recited in claim 60, wherein the authenticating
2 module determines an operating platform of the client.

3
4 62. A license server as recited in claim 60, further comprising:
5 a client image cache to store a set of client images; and
6 the client authenticating module receives a client image from the client and
7 compares the received client image to the set of client images to evaluate whether
8 the client is authentic.

9
10 63. A license server as recited in claim 60, wherein the granting module
11 encrypts the software license using a public key of the authenticated client.

12
13 64. A license server as recited in claim 60, wherein the software license
14 contains at least one of the following items:

15 a version indicator of a software license;
16 a license ID;
17 a client ID that identifies the authenticated client;
18 an issue date on which the license is issued to the client;
19 a platform type of the client's operating platform for which the software
20 license can be used;
21 an expiration date on which the software license will expire; and
22 a product ID that identifies a product with which the software licenses can
23 be used.

65. A license server as recited in claim 60, further comprising a license pack table to store information pertaining to the license pack that is stored in the license store.

66. A license server as recited in claim 60, further comprising a client assignment table containing a list of the software licenses that are granted to clients.

67. A license server as recited in claim 66, wherein the granting module upgrades the client assignment table after granting the software license to the authenticated client.

68. A program embodied on a computer-readable medium, comprising:
a code segment to receive a license pack from a license generator, the license pack containing a set of one or more individual software licenses;
a code segment to validate the license pack;
a code segment to store the software licenses;
a code segment, responsive to a request for a software license from a client, to determine whether the client is authentic and can receive a software license;
a code segment to grant a software license to an authenticated client, the software license containing a license ID; and
a code segment to associate the license ID with the authenticated client.

69. A client computer, comprising:
a license cache to store one or more software licenses;

1 a license requestor to request a software license from a license server;
2 a challenge handler to handle an authenticity challenge from the license
3 server, the challenge handler computing a challenge response that contains a client
4 image that can be used by the license server to evaluate whether the client is
5 authentic and can be licensed; and

6 whereupon authentication by the license server and granting of a software
7 license, the license requestor receiving the software license from the license server
8 and storing the software license in the license cache.

9
10 70. A client computer as recited in claim 69, wherein the challenge
11 contains a random number, and the challenger handler computes the challenge
12 response by concatenating the random number with the client image to form a
13 concatenated value and hashing the concatenated value.

14
15 71. A program embodied on a computer-readable medium, comprising:
16 a code segment to receive an authenticity challenge from a license server
17 that distributes software licenses;

18 a code segment to compute a challenge response that contains a client
19 image that can be used by the license server to evaluate whether the client is
20 authentic and can be licensed; and

21 a code segment to store the software license granted by the license server in
22 an event that the client is deemed authentic.

1 72. A data structure embodied on a computer-readable media,
2 comprising:

3 a license pack table to record information pertaining to one or more license
4 packs, the license pack table being indexed by license pack IDs that identify
5 corresponding individual license packs, each license pack containing one or more
6 software licenses;

7 a client assignment table to record information pertaining to software
8 licenses that are assigned to clients, the client assignment table being indexed by
9 license IDs that identify individual software licenses, the client assignment table
10 further having the license pack IDs of the license packs from which the
11 corresponding software licenses are issued; and

12 the license pack table and the client assignment table being correlated via
13 the license pack IDs contained in each table.

14
15 73. A data structure as recited in claim 72, wherein the license pack
16 table contains the following table fields:

17 a license pack ID field to hold the license pack ID;

18 a quantity field to hold a number representative of how many software
19 licenses are contained in the license pack;

20 a platform type field to hold a type of operating platform for which the
21 software licenses in the license pack can be used;

22 an expiration date field to hold a date on which the software licenses in the
23 license pack will expire; and

24 a product ID field to hold a product ID that identifies a product with which
25 the software licenses in the license pack can be used.

1
2 74. A data structure as recited in claim 73, wherein the license pack
3 table also contains a number assigned field to hold a number representative of how
4 many of the software licenses have been assigned to clients.
5

6 75. A data structure as recited in claim 72, wherein the client assignment
7 table contains the following table fields:

8 a license ID field to hold the license ID;

9 a license pack ID field to hold the license pack ID;

10 a client ID to hold an identifier of a client to which the software license is
11 granted; and

12 an issue date to hold a date on which the software license is issued to the
13 client.
14

15 76. A license pack data structure embodied on a computer-readable
16 media comprising:

17 a license pack ID field to hold an identifier for an associated license pack
18 that contains software licenses to be individually granted to individual computers;

19 a quantity field to hold a number representative of how many software
20 licenses are contained in the license pack;

21 a begin serial number to hold a beginning serial number of the software
22 licenses contained in the license pack;

23 an expiration date field to hold a date on which the software licenses will
24 expire; and
25

1 a product data field to hold data regarding the product with which the
2 software licenses can be used.

3
4 77. A license data structure embodied on a computer-readable media
5 comprising:

6 a version field to hold a version indicator of a software license;

7 a license ID field to hold an identifier for the software license;

8 a client ID to hold an identifier of a client to which the software license is
9 granted;

10 an issue date to hold a date on which the license is issued to the client;

11 an expiration date field to hold a date on which the software license will
12 expire; and

13 a product data field to hold data regarding the product with which the
14 software licenses can be used.

15
16 78. A license data structure as recited in claim 77, further comprising a
17 description field to hold a written description of the software license.

1 **ABSTRACT**

2 A software licensing system includes a license generator located at a
3 licensing clearinghouse and at least one license server and multiple clients located
4 at a company or entity. When a company wants a software license, it sends a
5 purchase request (and appropriate fee) to the licensing clearinghouse. The license
6 generator at the clearinghouse creates a license pack containing a set of one or
7 more individual software licenses. To prevent the license pack from being copied
8 and installed on multiple license servers, the license generator assigns a unique
9 license pack ID to the license pack and associates the license pack ID with the
10 particular license server in a master license database kept at the licensing
11 clearinghouse. The license generator digitally signs the license pack and encrypts
12 it with the license server's public key. The license server is responsible for
13 distributing the software licenses from the license pack to individual clients.
14 When a client needs a license, the license server determines the client's operating
15 system platform and grants the appropriate license. To prevent an issued license
16 from being copied from one client machine to another, the software license is
17 assigned to a specific client by including a client ID within the license. The
18 software license also has a license ID that is associated with the client ID in a
19 database record kept at the license server. The license server digitally signs the
20 software license and encrypts it using the client's public key. The license is stored
21 locally at the client.

1 **IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

2 Application Serial No.Not Assigned Yet
3 Filing DateNovember 28, 2000
4 Inventor..... Pradyunma K. Misra et al.
5 Group Art UnitNot Assigned Yet
6 ExaminerNot Assigned Yet
7 Attorney's Docket No.MS1-197USC1
8 Title: System and Method for Software Licensing

9 **REQUEST TO APPROVE DRAWING CHANGES**


10 To: Commissioner of Patents and Trademarks,
11 Washington, D.C. 20231

12 From: Allan T. Sponseller (Tel. 509-324-9256; Fax 509-323-8979)
13 Lee & Hayes, PLLC
14 421 W. Riverside Ave., Suite 500
15 Spokane, WA 99201

16 Enclosed is a photocopy of Fig. 3 of the drawings with proposed changes
17 indicated in red. Support for the changes to the drawings can be found, for
18 example, in the disclosure at page 17, lines 9-13. Applicant respectfully requests
19 that the proposed changes in Fig. 3 be approved by the Examiner.

20 Respectfully Submitted,

21 Date: 11/28/00

22 By: 
23 Allan T. Sponseller
24 Reg. No. 38,318
25 (509) 324-9256

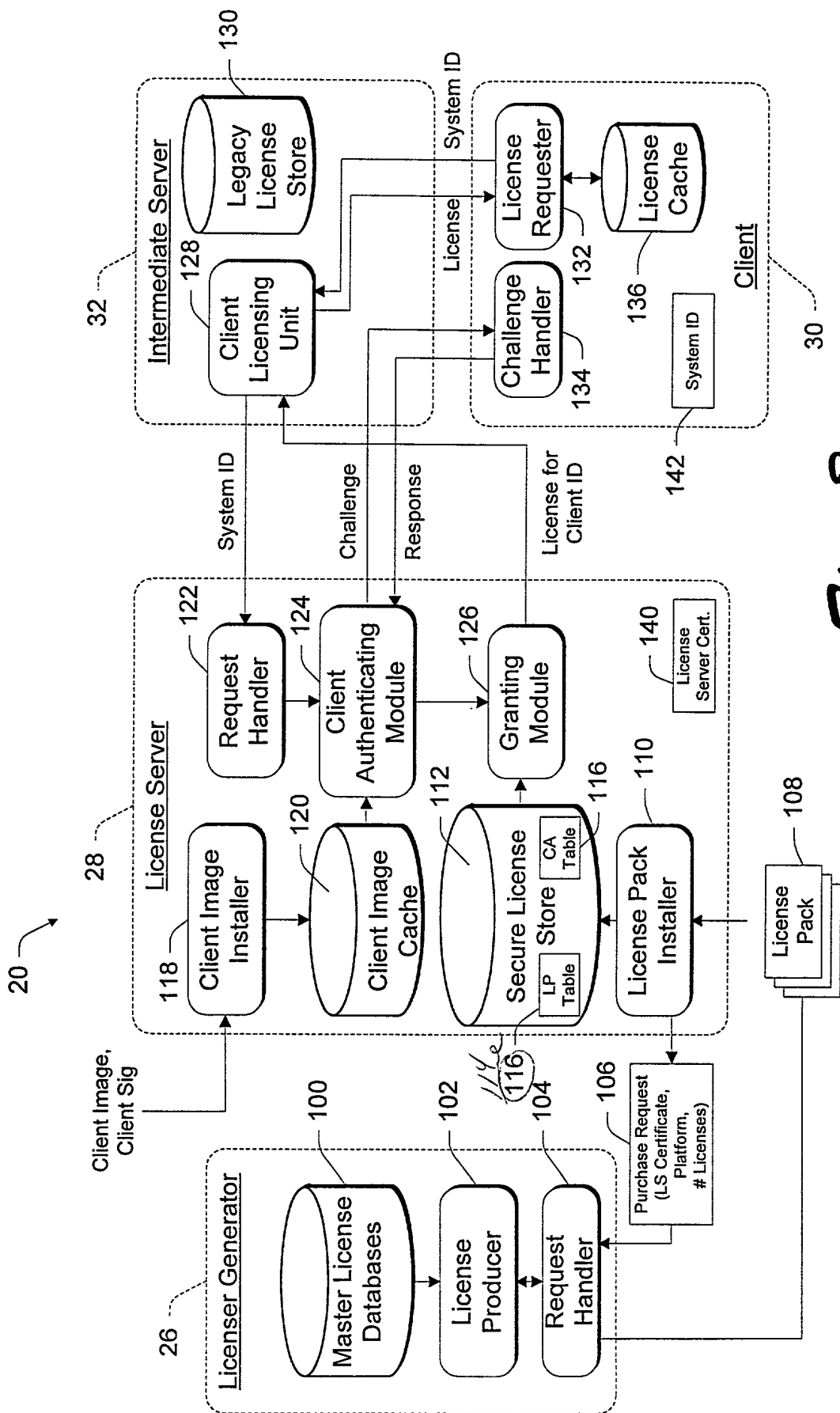
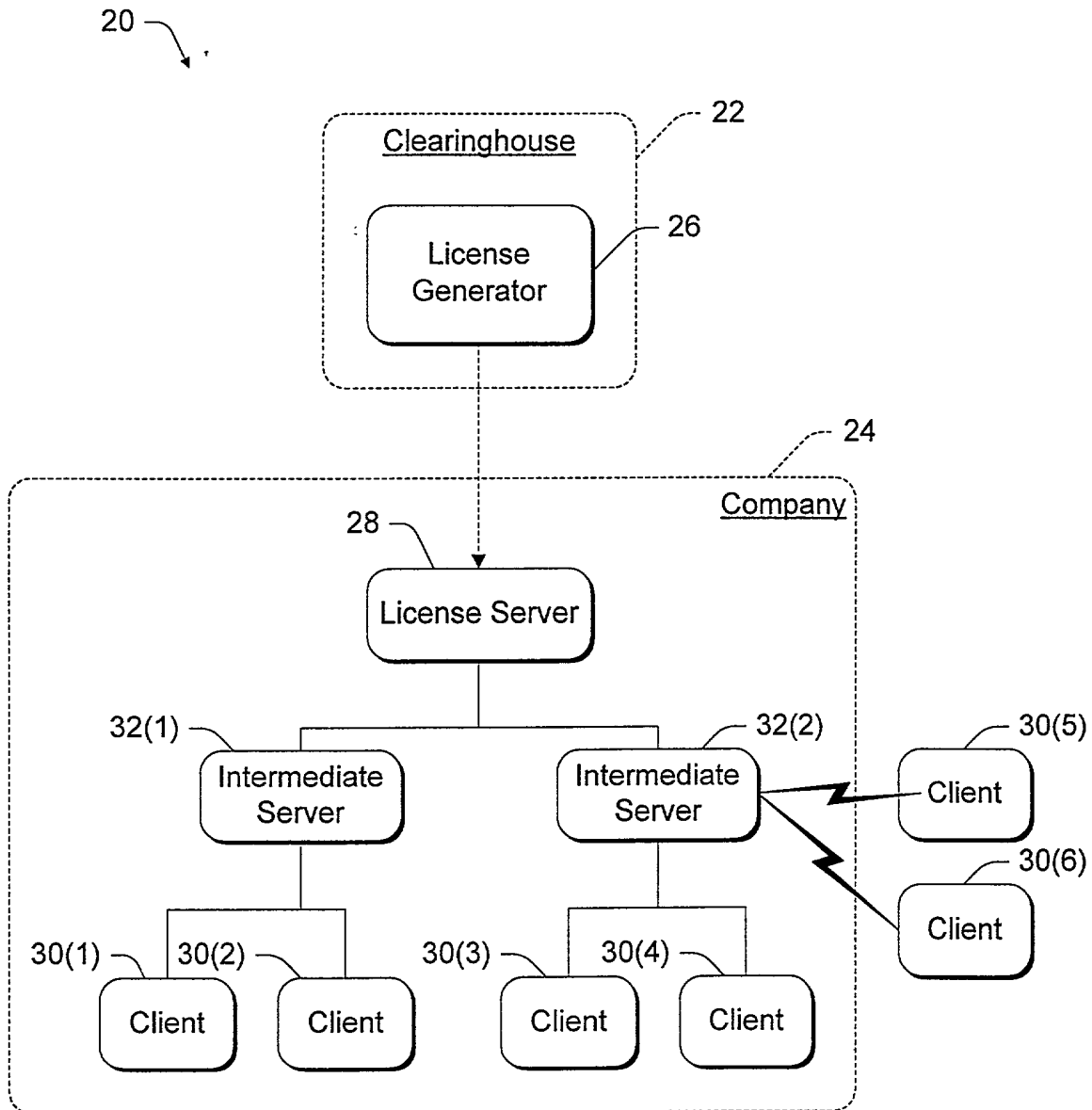
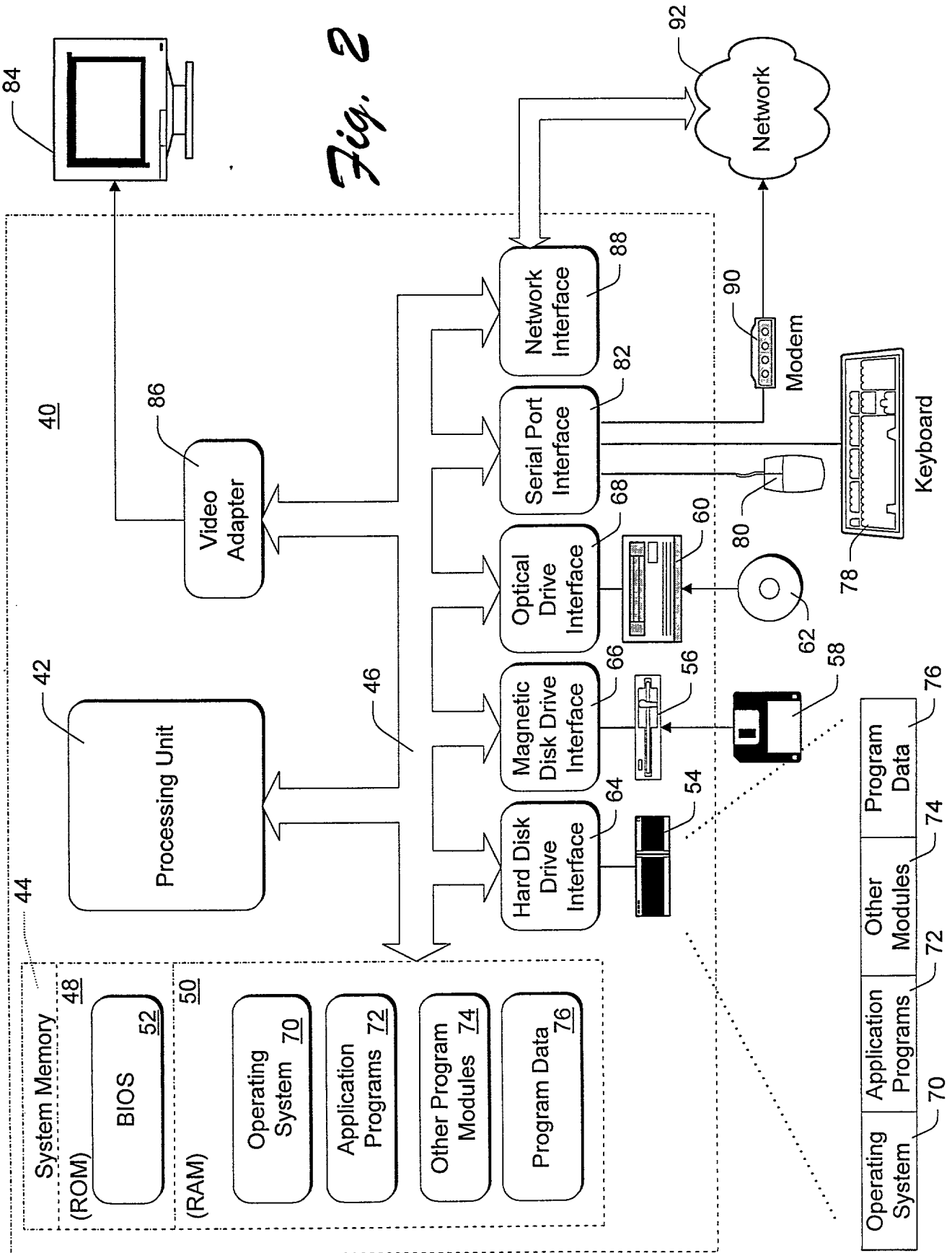


Fig. 3

*Fig. 1*

008277 8044260



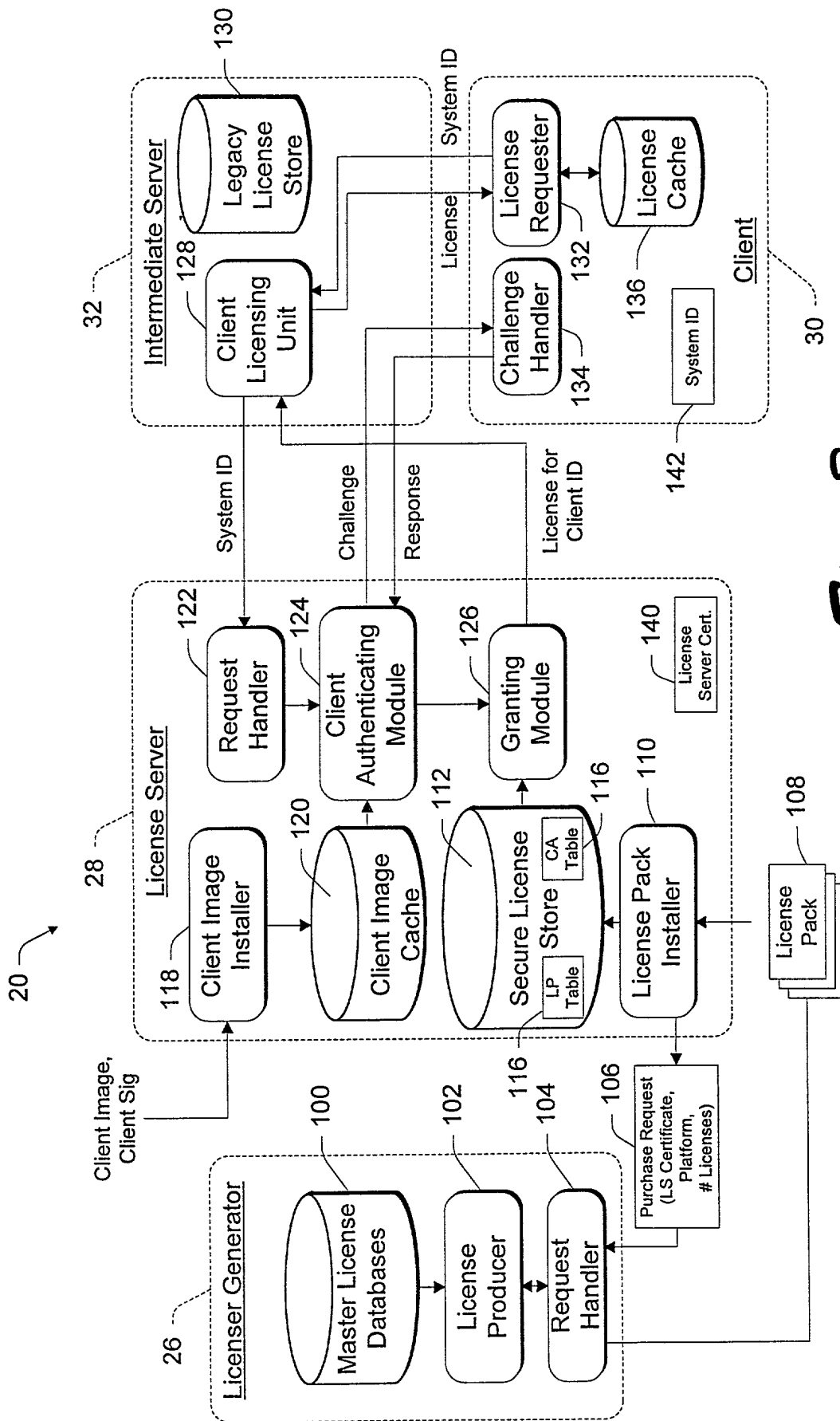


Fig. 3

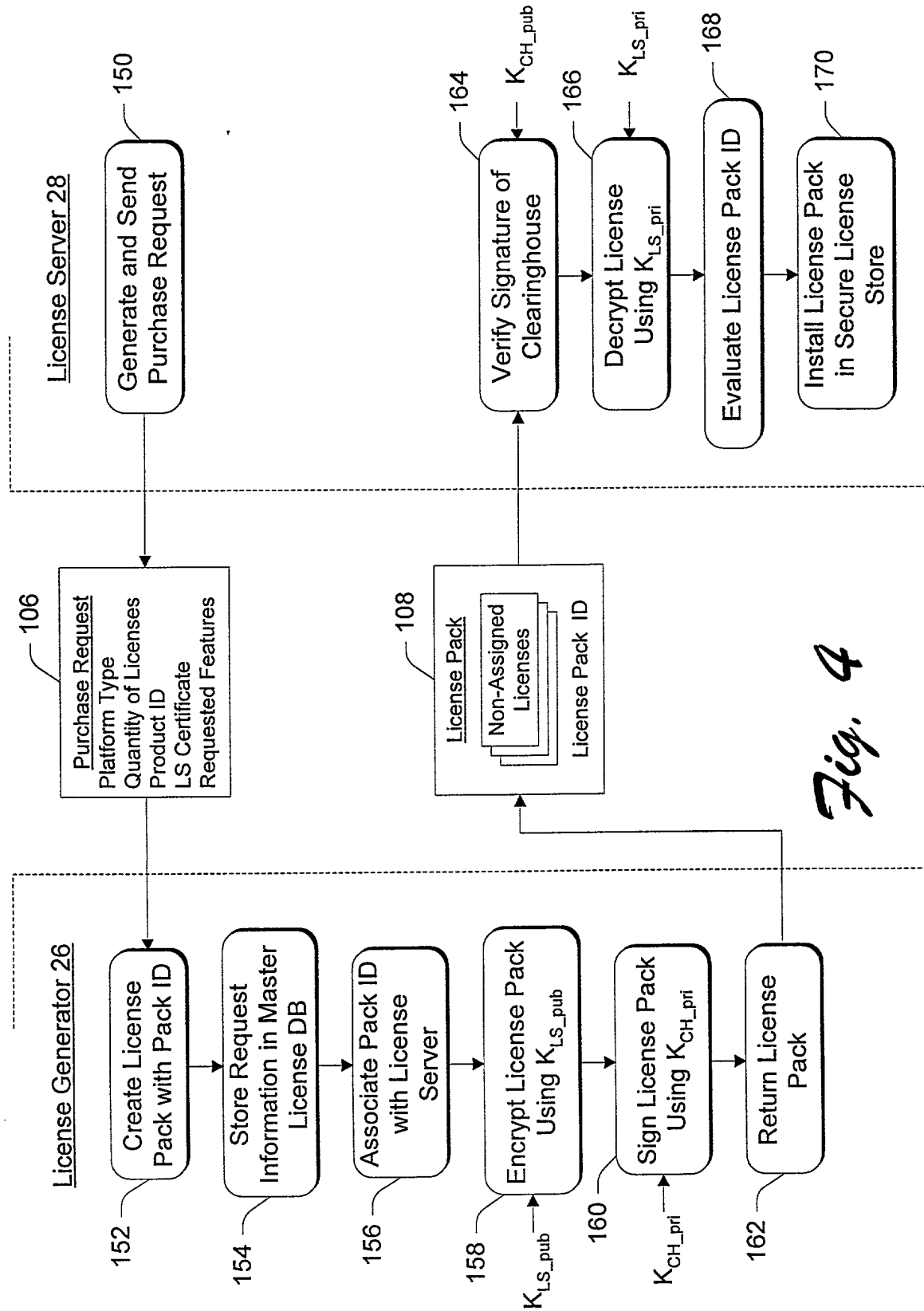
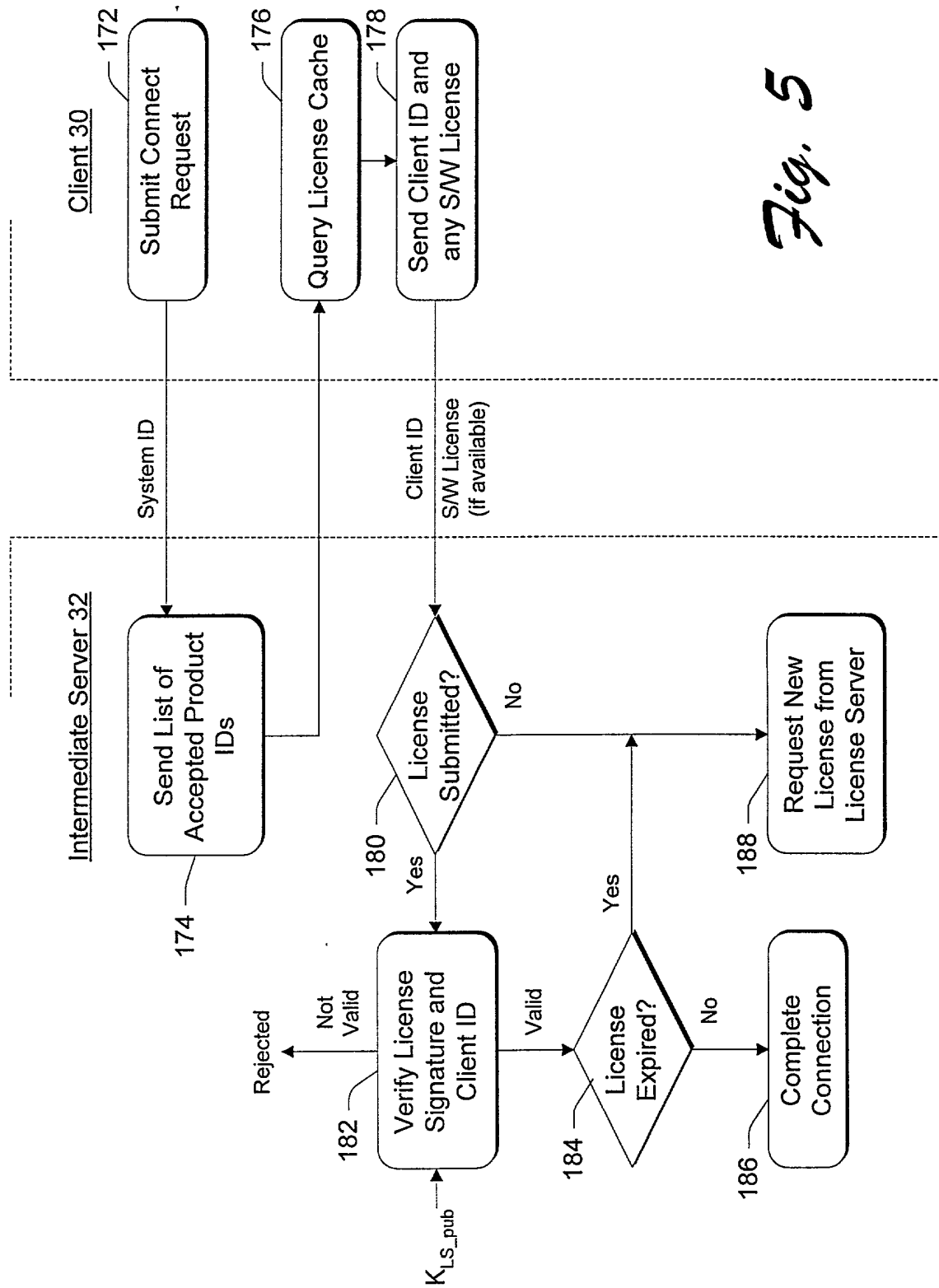
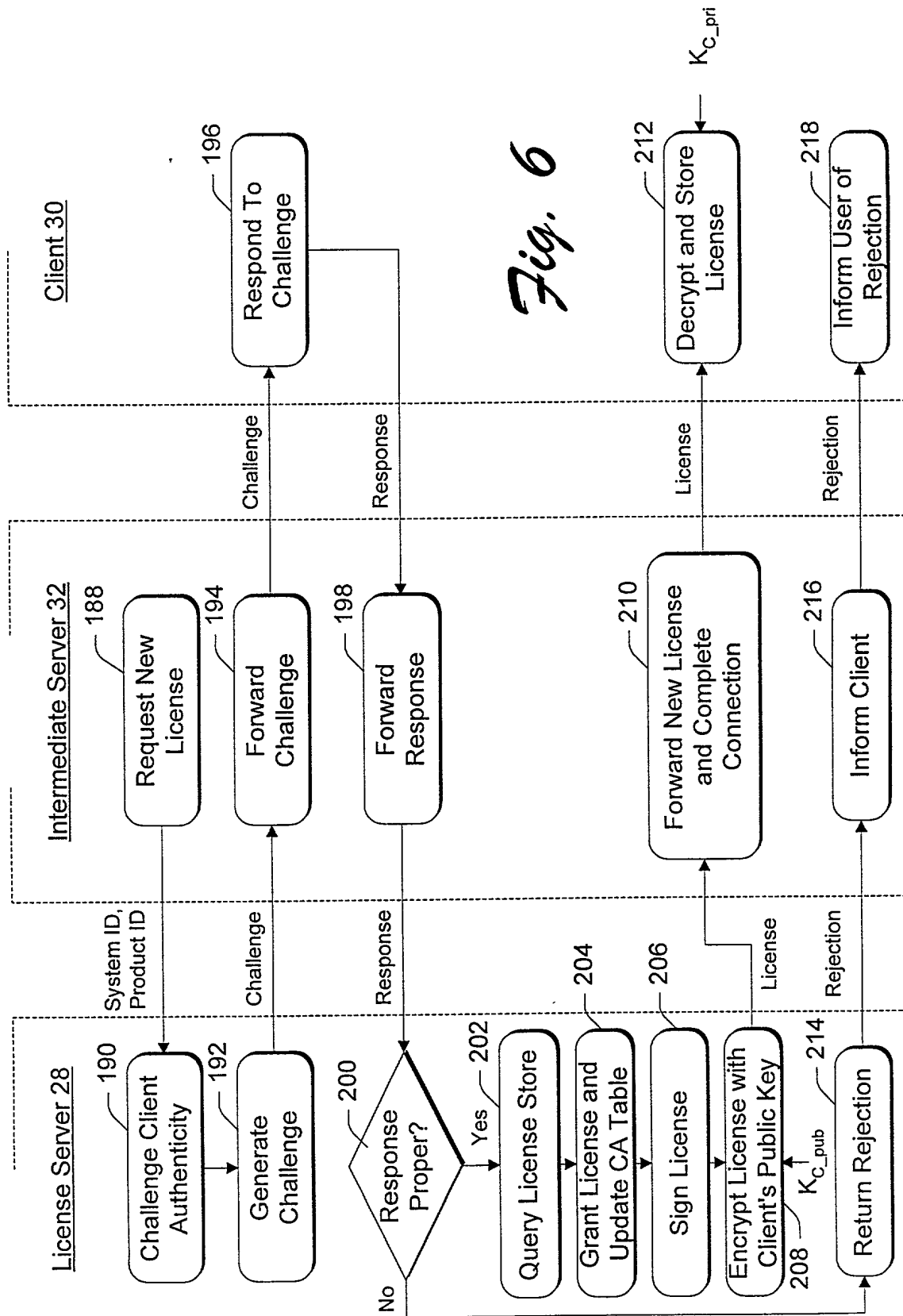


Fig. 4





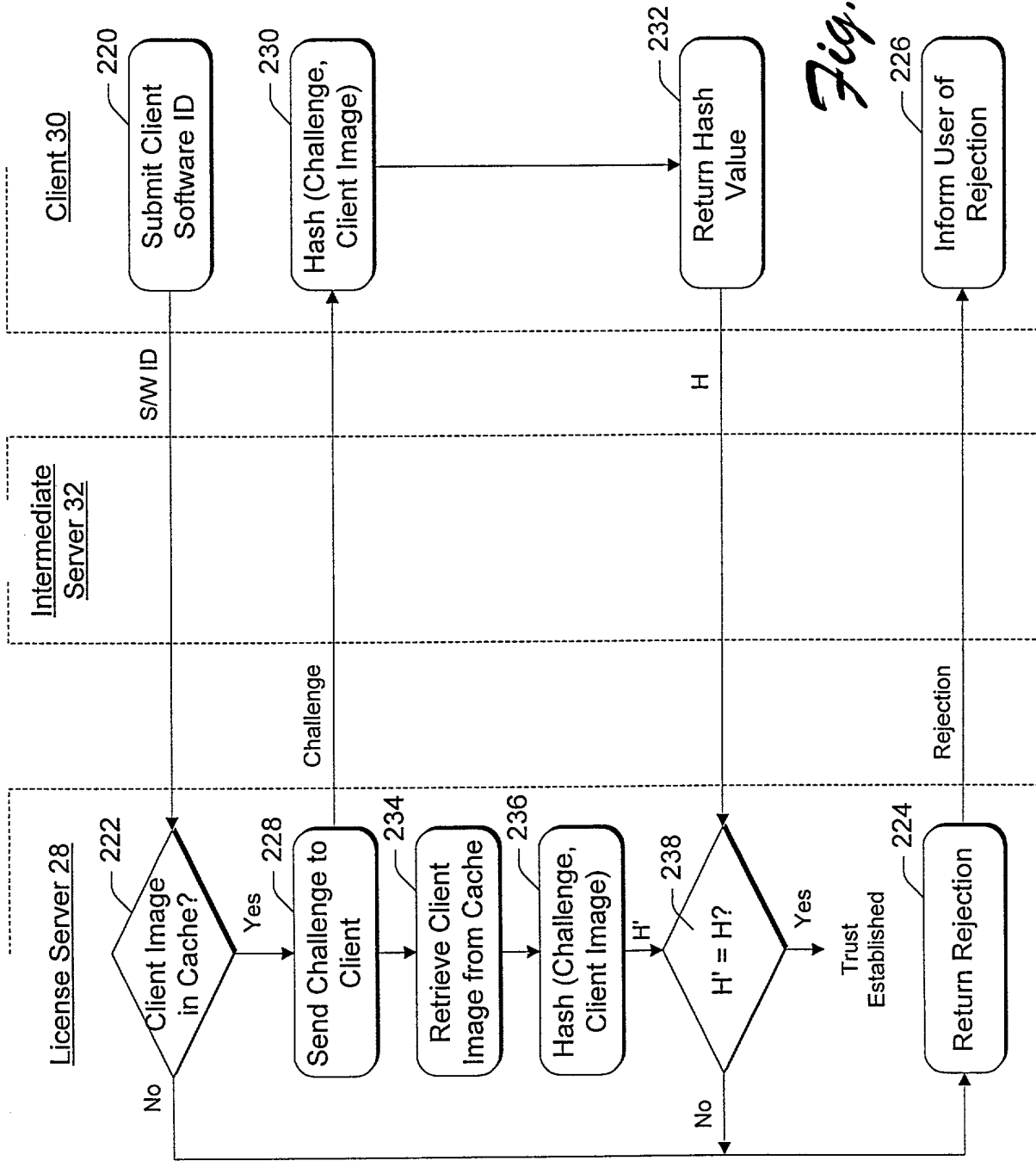


Fig. 7

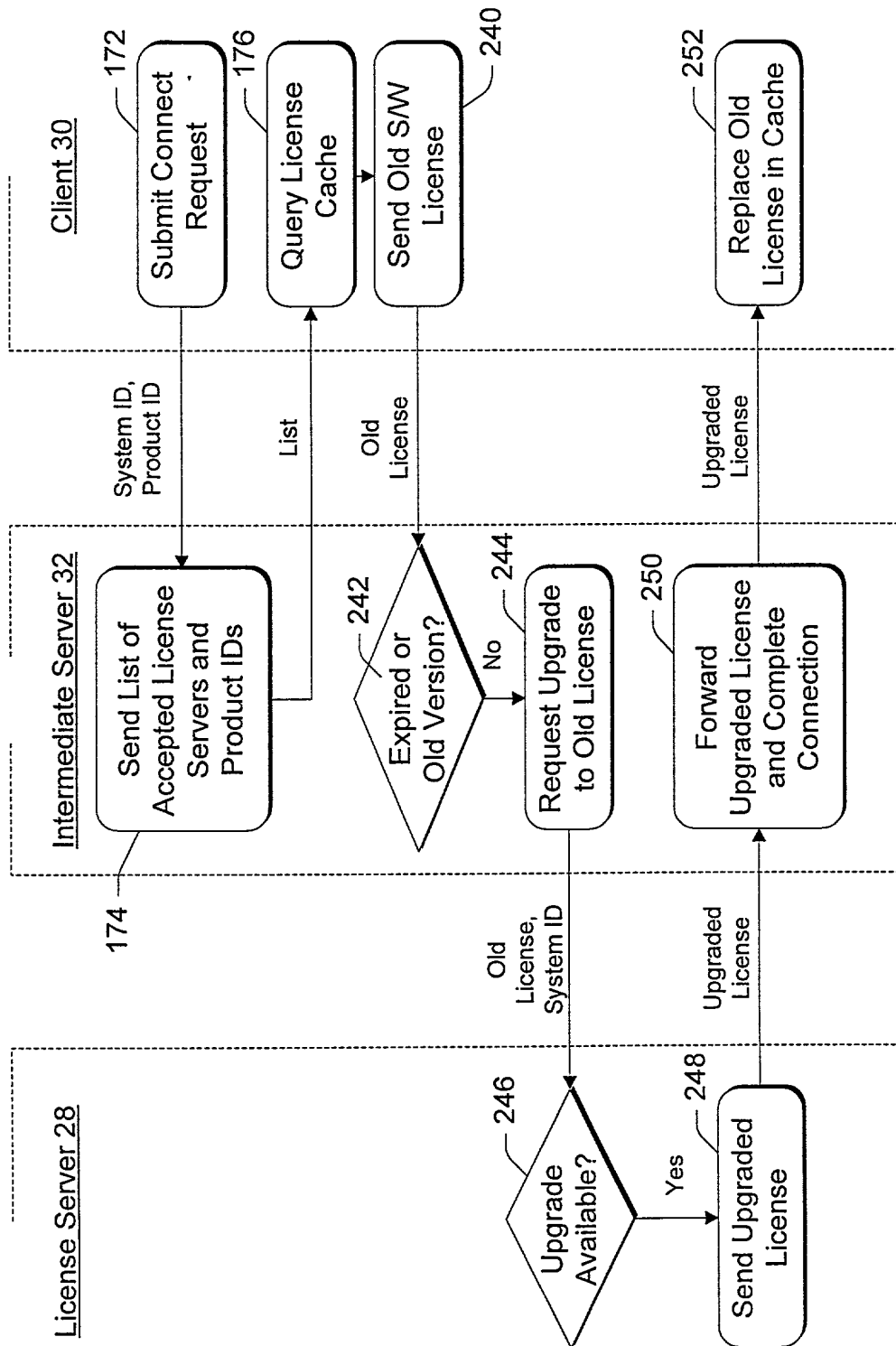


Fig. 8

1 **IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

2 Serial No.09/040,813
 3 Filing Date3/18/98
 4 InventorshipMisra et al.
 Applicant Microsoft Corporation
 Attorney's Docket No. MS1-197US
 Title: System and Method for Software Licensing

5 **DECLARATION FOR PATENT APPLICATION**

6 As a below named inventor, I hereby declare that:

7 My residence, post office address and citizenship are as stated below next to
 8 my name.

9 I believe I am the original, first and sole inventor (if only one name is listed
 10 below) or an original, first and joint inventor (if plural names are listed below) of the
 11 subject matter which is claimed and for which a patent is sought on the invention
 12 entitled "System and Method for Software Licensing" referenced above.

13 I have reviewed and understand the content of the above-identified
 14 specification, including the claims.

15 I acknowledge the duty to disclose information which is material to the
 16 examination of this application in accordance with Title 37, Code of Federal
 17 Regulations, § 1.56(a).

18 **PRIOR FOREIGN APPLICATIONS:** no applications for foreign patents or
 19 inventor's certificates have been filed prior to the date of execution of this
 20 declaration.

21 **Power of Attorney**

22 I appoint the following attorneys to prosecute this application and transact all
 23 future business in the Patent and Trademark Office connected with this application:
 24
 25

1 Lewis C. Lee, Reg. No. 34,656; Daniel L. Hayes, Reg. No. 34,618; Katie E. Sato,
2 Reg. No. 32,628 and Daniel D. Crouse, Reg. No. 32,022.

3 Send correspondence to: LEE & HAYES, PLLC, W. 201 North River Drive,
4 Suite 430, Spokane, Washington, 99201. Direct telephone calls to: Lewis C. Lee
5 (509) 324-9256.

6
7 All statements made herein of my own knowledge are true and that all
8 statements made on information and belief are believed to be true; and further that
9 these statements were made with the knowledge that willful false statements and the
10 like so made are punishable by fine or imprisonment, or both, under Section 1001 of
11 Title 18 of the United States Code and that such willful false statement may
12 jeopardize the validity of the application or any patent issued therefrom.

13
14 * * * * *

15 Full name of inventor: Pradyumna K. Misra

16 Inventor's Signature Pradyumna K. Misra Date: 5/27/98

17 Residence: Redmond, Washington

18 Citizenship: USA

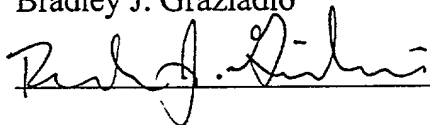
19 Post Office Address: 5020 159th Court NE
20 Redmond, WA 98052

003471 804260

Full name of inventor:

Bradley J. Graziadio

Inventor's Signature



Date: 27-May-99

Residence:

Redmond, Washington

Citizenship:

USA

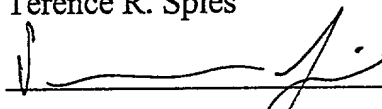
Post Office Address:

2827 233rd Place NE
Redmond, WA 98053

Full name of inventor:

Terence R. Spies

Inventor's Signature



Date: 3/27/98

Residence:

Kirkland, Washington

Citizenship:

USA

Post Office Address:

213 5th Ave. West
Kirkland, WA 98033

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Serial No.09/040,813
 Filing Date3/18/98
 InventorshipMisra et al.
 Applicant Microsoft Corporation
 Attorney's Docket No. MS1-197US
 Title: System and Method for Software Licensing

PATENT ASSIGNMENT**PARTIES TO THE ASSIGNMENT****Assignor(s):**

Pradyumna Misra
 5020 159th Court NE
 Redmond, WA 98052

Bradley J. Graziadio
 2827 233rd Place NE
 Redmond, WA 98053

Terence R. Spies
 213 5th Ave. West
 Kirkland, WA 98033

Assignee:

Microsoft Corporation
 Corporation of the State of Washington
 One Microsoft Way
 Redmond, WA 98052-6399

AGREEMENT

WHEREAS, Assignor(s) are inventor(s) of an invention entitled System and Method for Software Licensing," as described and claimed in the specification forming part of an application for United States letters patent referenced above;

WHEREAS, Microsoft, a corporation of the State of Washington having a place of business at One Microsoft Way, Redmond, WA 98052, is desirous of acquiring the entire right, title and interest in and to the invention and in and to any

1 letters patent that may be granted therefor in the United States and in any and all
2 foreign countries;

3 NOW, THEREFORE, in exchange for good and valuable consideration, the
4 receipt of which is hereby acknowledged, Assignor(s) hereby sell, assign and
5 transfer unto Microsoft, the entire right, title and interest in and to said invention,
6 said application and any and all letters patent which may be granted for said
7 invention in the United States of America and its territorial possessions and in any
8 and all foreign countries, and in any and all divisions, reissues and continuations
9 thereof, including the right to file foreign applications directly in the name of
10 Microsoft and to claim priority rights deriving from said United States application
11 to which said foreign applications are entitled by virtue of international
12 convention, treaty or otherwise, said invention, application and all letters patent on
13 said invention to be held and enjoyed by Microsoft and its successors and assigns
14 for their use and benefit and of their successors and assigns as fully and entirely as
15 the same would have been held and enjoyed by Assignor(s) had this assignment,
16 transfer and sale not been made. Assignor(s) hereby authorize and request the
17 Commissioner of Patents and Trademarks to issue all letters patent on said
18 invention to Microsoft. Assignor(s) agree to execute all instruments and
19 documents required for the making and prosecution of applications for United
20 States and foreign letters patent on said invention, for litigation regarding said
21 letters patent, or for the purpose of protecting title to said invention or letters
22 patent therefor.

5/27/98
Date Pradyumna K. Misra

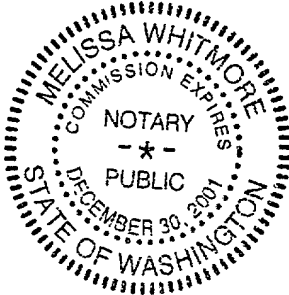
State of Washington)
County of King) ss.

I certify that I know or have satisfactory evidence is the person Pradyumna K. Misra who appeared before me, and said person acknowledged that he signed this instrument and acknowledged it to be his free and voluntary act for the uses and purposes mentioned in the instrument.

Dated May 27, 1998

Signature of Notary Public Melissa Whitmore

My appointment expires December 30, 2001



27-May-98
Date Bradley J. Graziadio

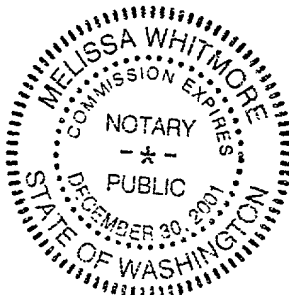
State of Washington)
County of King) ss.

I certify that I know or have satisfactory evidence is the person Bradley J. Graziadio who appeared before me, and said person acknowledged that he signed this instrument and acknowledged it to be his free and voluntary act for the uses and purposes mentioned in the instrument.

Dated May 27, 1998

Signature of Notary Public Melissa Whitmore

My appointment expires December 30, 2001



5-27-98

Date

Terence R. Spies

State of Washington

County of King

ss.

I certify that I know or have satisfactory evidence Terence R. Spies is the person who appeared before me, and said person acknowledged that he signed this instrument and acknowledged it to be his free and voluntary act for the uses and purposes mentioned in the instrument.

Dated May 27, 1998

Signature of Notary Public Melissa Whitmore

My appointment expires December 30, 2001

